1984

Image data compression using uneven knots selection.

De Miranda, Cesar Taedu

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMAGE DATA COMPRESSION
USING UNEVEN KNOTS SELECTION

by

Cesar Tadeu de Miranda

June 1984

Thesis Advisor: Chin-Hwa Lee

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

Image Data Compression Using Uneven Knots Selection

5. TYPE OF REPORT & PERIOD COVERED

Master's/Engineer's Thesis, June 1984

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Cesar Tadeu de Miranda

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93943

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS

Naval Postgraduate School
Monterey, California 93943

12. REPORT DATE

June 1984

13. NUMBER OF PAGES

251

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Processing, Data Compression, Uneven Knot Selection, B-Cubic Spline, Scattered Data Interpolation, Thin Plates Spline.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Application of B-spline functions over uneven knots in image processing is studied here. The subject is on compression of image signals. The algorithm for selecting uneven knots from the contours of the image is pursued here. A specific closeness error constraint appropriate for 3-D data is used here. The image reconstruction problem is tackled in two distinct ways, namely triangulation method and thin plates spline method. Some procedures to evaluate the fidelity of the image reconstruction schemes are provided too. A set of experiments in a broad variety of aerophotograph images is performed in order to test and validate the overall algorithm.

Approved for public release, distribution unlimited

Image Data Compression Using Uneven Knots Selection

by

Cesar Tadeu de Miranda
Captain, Brazilian Air Force
B.S., Instituto Tecnologico de Aeronautica, 1977

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
AND ELECTRICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
June 1984

ABSTRACT

Application of B-spline functions over uneven knots in image processing is studied here. The subject is on compression of image signals. The algorithm for selecting uneven knots from the contours of the image is pursued here. A specific closeness error constraint appropriate for 3-D data is used here.

The image reconstruction problem is tackled in two distinct ways, namely triangulation method and thin plates spline method. Some procedures to evaluate the fidelity of the image reconstruction schemes are provided too. A set of experiments in a broad variety of aerophotograph images is performed in order to test and validate the overall algorithm.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to express my gratitude to everybody who direct or indirectly contributed for the accomplishment of this thesis.

To the 130 millions of Brazilian tax payers who provided my financial support.

To my parents, Luiz and Maria Miranda, who dedicated all their lives in my education.

To my Thesis Advisor, Dr. Chin-Hwa Lee, for his advice, assistance and patience.

To Dr. Mitchel Cotton, Dr. Richard Franke, and Dr. Paul Dierckx for their contributions.

Finally, I am grateful to my wife, Maria Cristina, for her extraordinary help in the typing, spelling and suggestions of this work.

I. INTRODUCTION

A. BACKGROUND

Interests in digital image processing methods comes from two principal areas: improvement of pictorial information for human interpretation, and processing of scene data for autonomous machine perceptions.

Since the second decade of this century the interest for image processing has become strong, although the theory was available at that time, the absence of processing power (speed and bulk storage) delayed the development. Only in the earliers 60's, with the revolution of the semiconductor industry which speeded up the computers and at the same time reduced drastically the storage media cost, the image processing techniques flourished.

Today fast devices and virtual memory computers allow more efficient and simple implementation of algorithms without the "complex" and nonuser friendly environment like overlaying and segmentation of the image.

The Figure 1.1 shows a typical image processing system. The fundamental operations of such a system fall into four major categories: digitization, processing, storage, and display.

B. DIGITAL IMAGE REPRESENTATION

The image signals can be represented in two distinct groups. The color image and the monochromatic image which will be referred throughout this work.

Any generic image (monochromatic) can be represented by a function f(x,y), where x and y denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness (or gray level) of the image at that point.

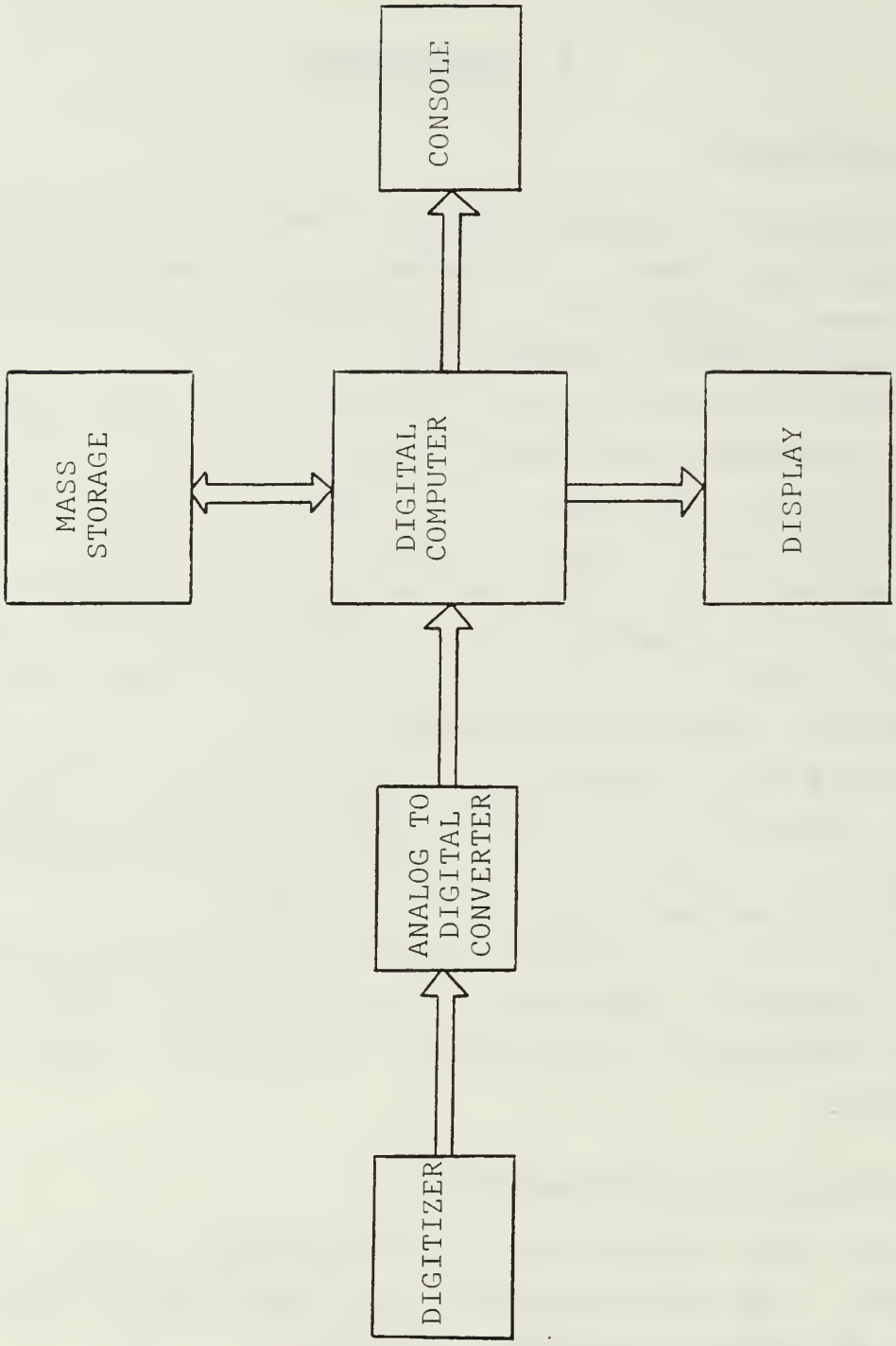

Figure 1.1 Typical Image Processing System.

As far as digital processing and storage are concerned we need to have a digital representation of images. The image f(x,y) which has been discretized in spatial coordinates referred to as image samples, is quantized as gray levels. Therefore, a digital image can be represented as values at grid (that may be uniform or nonuniform).
The rows and columns indices locates the quantized gray level at that point. Each element in this grid is called a picture element or simply pixel.

Although the number of rows, columns, and gray levels is arbitrary, it is a good policy to work with power of 2^n due to the simple reason that we are processing images in a binary machine.

Throughout this work, a standard (256x256) square grid will be used to represent even sampled images with the gray level quantized in the range from zero to 255.

C. IMAGE PROCESSING TECHNIQUES

1. Introduction

Basically, the techniques for digital image processing can be classified into four main categories:

Image digitization
Image enhacement
Image encoding
Image segmentation and representation

The above techniques can be implemented in two different and highly competitive approaches: the space domain approach and the transform domain approach. Both have advantages and disadvantages, and if we consider the image processing as a whole, there is no clear boundary between each other.

For a particular application, i.e., depending on the specific work we have to do, and the set of data we have available, we can select the approach among the space or transform domain techniques [Ref. 1].

2. Spatial Domain Techniques

Spatial domain techniques are based on gray level mapping. The mapping used depends on the kind of the image processing that we are willing to do. Gonzales, [Ref. 1] describes these mapping techniques in a comprehensive way. Histogram equalization, contrast stretching, smoothing, sharpenig, filtering, contour encoding are classical space domain techniques. More recently [Ref. 2] there are some new ones, like image erosion and image dilation, which can be combined to generate opening and closing effects in the image.

The mathematical basis for these techniques are algebraic manipulations such as vector and matrix operations.

3. Transform Domain Techniques

Among the transform domain representations the important ones are Fourier transform, Walsh transform, Hadamard transform, singular value decomposition (Hotelling transform), Kornhuen Loeve transform, etc. [Ref. 3]. These transforms use different discrete basis functions such as Fourier basis, cosine basis, Hadamard basis, to reveal certain features of the signal. The assumptions which support these methodologies are that the system is linear, shift invariant, and that the signal is uniformly sampled. This last constraint, as we will see in this text, is quite strong drawback for the transform methods in some applications.

Since these transform domain representations are derived from the basic space domain uniformly sampled gray levels of the signal, the accuracy of these transforms domain representations are only as good as that of uniform sampled gray level itself. Therefore, like the traditional space domain representation, the transform domain ones can also be viewed as a linear approximation of the original signal by using different basis functions.

4. Nonconventional Techniques

In order to overcome some difficulties of the traditional approches for image processing, such as the uniform sampling constraint, it is desirable to seek an alternative signal representation from which employs a complex basis functions for better efficiency and achieves a higher compression ratio through the notion of function approximations.

The idea to use nonuniform sampling of image signal is based on the fact that in overall image pixels have quite localized correlation, but over a small region the correlation is usually high. Therefore, an adaptive sampling technique which allows us to have more sampling points in busy areas and less ones in doll areas seems to be interesting. An approach like this allow us to avoid oversampling in low frequency regions and undersampling in high frequency regions, therefore, reducing the number of necessary pixels to represent the image (that is, achieve data compression) with adequate accuracy.

As we will see later that the polynomial spline, particularly the B-spline functions, are chosen for such alternative representation.

These B-spline basis have a local behavior that seems to match the local correlation of the image pixels. As we will see, there is a tradeoff between the mathematical

complexity of this new image representation (B-spline) form and its efficient and powerful localized behavior.

D. PROBLEM DESCRIPTION

1. Motivations

Since the image signal requires a huge bulk storage capability and a wide bandwidth during communications, the encoding techniques are a natural way to minimize this problem.

In order to illustrate the problem some examples will be given:

i) A typical aero-photographic image frame (256x256) over square grid will be dealt with throughout this work. It requires about 64k of memory, that is, the total memory capability of the majority of the microprocessors available today.

ii) The transmission of one of the above frame, a line with 1200 baud rate for instance, requires about 50 seconds. That is extremely long for most applications.

iii) The LANDSAT images are stored on magnetic tapes [Ref. 1] where the number of bits required to store one frame is (2340)(7)(4) = 211000000. The LANDSAT collects 30 frames per day or approximately $2.2\ 10^{11}$ bit per year. If the data (number of bits) could be reduced by a factor of two, the number of data tapes (and the number of warehouses required to store them) could be reduced by the same factor.

2. Objectives

Based on the rationale mentioned in the previous sections, the objective of this work is to explore the advantages of image encoding, more specifically data compression, by using a nonorthodox technique. The problem of uneven sampling representation will be pursued.

The mathematical framework is based on the polynomial splines functions, more specifically the B-splines basis, which have shown recently [Ref. 4] to be quite suitable for digital image representation, restoration, and interpolation.

3. Formal Specification of the Problem

Given a digital image represented by its gray levels, we are asked to code this image in contours. From these contours, we select uneven knots based on a criterion that gives maximum smoothness under the constraint of closeness. These requirements should take into account the three-dimensional effects of the image.

In other words, it may be stated that the main goal is that the number of selected knots should be less so that a high data compression ratio is achieved without introducing large errors in the reconstructed image.

In the next chapter the concept and definition of knot and error measurements will be introduced.

II. SPLINE FUNCTIONS AND B-SPLINES

A. INTRODUCTION

The objective of this chapter is to present an overview of the spline functions theory that is the basis of the development of the next chapters. This technique was first formally introduced as a "mathematical function" by Schoenberg in 1946 and it has become an important tool in various applications, such as, approximation, interpolation, and curve-fitting to a set of discrete data points.

Some of the reasons for the great popularity of the spline functions are because they are simple and yet have many desirable characteristics in the above mentioned applications. Particularly, the representation of a piecewise spline function through a linear combination of normalized B-spline basis has some interesting characteristics which relates very well to the image model that was seen in Chapter I.

B. SPLINE DEFINITION

The name spline derives from a draftman's device. A spline is a flexible strip which can be bended by weights, so that it passes through each given point. It also goes smoothly from each interval to the next, according to the law of beam flexure. The mathematical procedure presented here is an extension of this idea.

A spline function is composed of piecewise polynomials satisfying certain continuity properties at the end points. As we can see this is a loose definition. There are several formal definitions for the spline function, [Ref. 3], [Ref. 6], [Ref. 10], [Ref. 16], although the basic idea of

the informal definition remains the same. Since we will be using a modifying version of Dierckx's algorithm in Chapter III, his notation is adopted. Therefore, the function spline is formally defined as follow: a function S(x), defined in the range $a \leq x \leq b$, is called a spline function of degree K with knots t_i, $i = 1,2,\ldots,n$ if the following conditions are satisfied:

(i) In each interval $[t_{i-1}, t_i]$, $i = 1,2,\ldots,n+1$ ($a = t_0$, $b = t_{n+1}$), S(x) is given by a polynomial of degree K or less.

(ii) S(x) and its derivatives of order $1,2,\ldots,K-1$ are continuous everywhere in the interval [a,b].

The order of the spline function is defined as

$$m = k + 1 \tag{2.1}$$

where m is the order of the spline and k is the degree of the spline.

Example: For a spline of degree K = 3 (cubic spline) the order of the spline is m = 4; therefore, we can see that the degree of the spline is the high power of the polynomial, and the order of the spline is the number of coefficients in the complete polynomial spline.

The sequence $(t_i)_1^n$ is called a knot sequence (also known as joint sequence or break point sequence for our application here).

There is a generalized definition of a spline function not only that makes use of the maximum smoothness allowed, but it uses piecewise polynomials with less then K-1 continuous derivatives, which are known as deficient splines.

By using this generalized definition, the break point sequence (or joint sequence) will not be the same as the knot sequence anymore.

A break point sequence is a sequence of strictly increasing real numbers $t_1 < t_2 \ldots < t_n$ whereby each break point t_i is the common joint between two adjacent polynomial pieces.

The knot sequence is a monotonic increasing sequence of real numbers $x_0 \leq x_1 \leq \ldots \leq x_m$ $(m \geq n)$, such that one or more knots (x_j's) may be collapsed into a single break point (t_i).

The advantage of this extended definition is that it can represent better abrupt transitions and even jumps in the function. The less continuous is the function in a given break point more knots will collapse into that point as shown in Figure 2.1.

As we will see in Chapter III, we are interested in generating a smooth curve through a set of contour data points; therefore, we will use the restrict definition so that the break point sequence coincides with the knot sequence, and they will be interchangeable.

C. CUBIC SPLINE FUNCTION

According to the definition adopted here, the degree of the spline function is an arbitrary integer number K. However, a piecewise low degree polynomial is usually more suitable to form a curve through a series of points. The advantages in using low-degree polynomials are that it reduces the computational requirements and avoids the numerical instabilities that arises with high degree polynomials. These instabilities can cause undesirable wiggles when many points must be joined in a common curve. On the other hand, however, the low order spline such as the first and second order splines, i.e., the pulse function and the piecewise linear function respectively, are not smooth. For most of the applications they are not suitable either. Therefore, we

have a tradeoff between low and high order spline functions, based on the above considerations. The cubic spline (degree K = 3 and order m = 4) is advantageous since it is the lowest degree space curve which allows a point of inflection and has the ability to twist through space. Therefore, the cubic spline function is a natural choice, and it will be used from now on.

The cubic spline function $S_n(t)$ defined over a set of knots (or break points), $(t_i)_0^n$, is represented in each interval (t_i, t_{i+1}) by a cubic polynomial function.

$$S(t) := y = a_i (t-t_i)^3 + b_i (t-t_i)^2 + c_i (t-t_i) + d_i \qquad (2.2)$$

for $t_i < t < t_{i+1}$.

Since it fits at the two ends of the interval,

$$y_i = a_i (t_i - t_i)^3 + b_i (t_i - t_i)^2 + c_i (t_i - t_i) + d_i = d_i \qquad (2.3)$$

$$y_{i+1} = a_i (t_{i+1} - t_i)^3 + b_i (t_{i+1} - t_i)^2 + c_i (t_{i+1} - t_i) + d_i \qquad (2.4)$$

$$= a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i$$

where $h_i = t_{i+1} - t_i$.

In order to examine the slope and the curvature of the joined polynomials, we need the first and the second derivatives. So by differentiating equation 2.2 we get:

$$y' = 3a_i(t-t_i)^2+2b_i(t-t_i)+c_i \qquad (2.5)$$

$$y'' = 6a_i(t-t_i)+2b_i \qquad (2.6)$$

Now, if we define the second derivative y'' at (t_i, y_i) and (t_{i+1}, y_{i+1}) by S_i and S_{i+1} respectively, we can express three coefficients of the polynomial in terms of the given points (t_i, y_i) and the second derivative S_i of the curve at each point.

(2.7)

$$d_i = y_i$$

$$c_i = \frac{y_{i-1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

$$b_i = S_i/2$$

$$a_i = (S_{i+1} - S_i)/2$$

Now, from the condition that the slopes of two cubics which join at (t_i, y_i) are the same, from equation 2.5 at the points t_i and t_{i-1} we have

$$y_i' = 3a_i(t_i-t_i)^2+2b_i(t_i-t_i)+c_i \qquad (2.8)$$

$$y_i' = 3a_{i-1}(t_i-t_{i-1})^2+2b_{i-1}(t_i-t_{i-1})+c_{i-1} \qquad (2.9)$$

Equating equations 2.8, 2.9 and pluging the coefficients expressed in equation 2.7 we have the general term

$$h_{i-1} S_{i-1} + 2(h_{i-1} + 2h_i) + h_i S_{i+1} = \tag{2.10}$$

$$= \left(\frac{6\ y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right)$$

The equation 2.10 applies for all given points i except the boundaries i = 0 and i = n. Therefore, if we write equation 2.10 in matrix form for i = 1,2,...,n-1 where S_i are the unknown we have

(2.11)

$$\begin{bmatrix} h_1 & 2(h_1 + h_2) h_2 & & 0 \\ & h_2 & 2(h_2 + h_3) h_3 & \\ & & \cdot & \\ & & \cdot & \\ & & \cdot & \\ 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ \cdot \\ S_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_n \end{bmatrix}$$

with

$$b_i = 6 \left\{ \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}} \right\}$$

The above matrix has n-2 equations and n unknowns. The two extra equations are given from the end conditions which may be given in different form depending upon the specific problem, as we will see.

Furthermore, the above matrix becomes tridiagonal and as its properties are well known; it can be solved fast and can be stored economically.

The end contitions characterize the different types of cubic spline. The common cases are listed below.

(i) Natural spline

$$S'(t_0) = S''(t_n) = 0 \tag{2.12}$$

In spite of its positive sound name, the natural spline due to its flatter effects at ends, is not recommended from an approximation theoretic point of view [Ref. 10].

(ii) P-spline

$$S_0'(t) = K_1 \text{ and } S_{n-1}'(t) = K_2 \tag{2.13}$$

where K_1 and K_2 are constant and the first and last span are covered by a parabola.

(iii) Cyclic or periodic spline

$$S^{(j)}(t_0) = S^{(j)}(t_n) \qquad j = 0,1,2 \tag{2.14}$$

This is quite appropriate to represent closed curves as we will see in Chapter III.

(iv) Anti-cyclic spline

$$S^{(j)}(t_0) = S^{(j)}(t_n) \qquad j = 0,2 \tag{2.15}$$

$$S^{(j)}(t_0) = -S^{(j)}(t_n) \qquad j = 1$$

(v) Canti-levered spline

$$S'(t_0) = S'(t_n) = 0 \qquad (2.16)$$

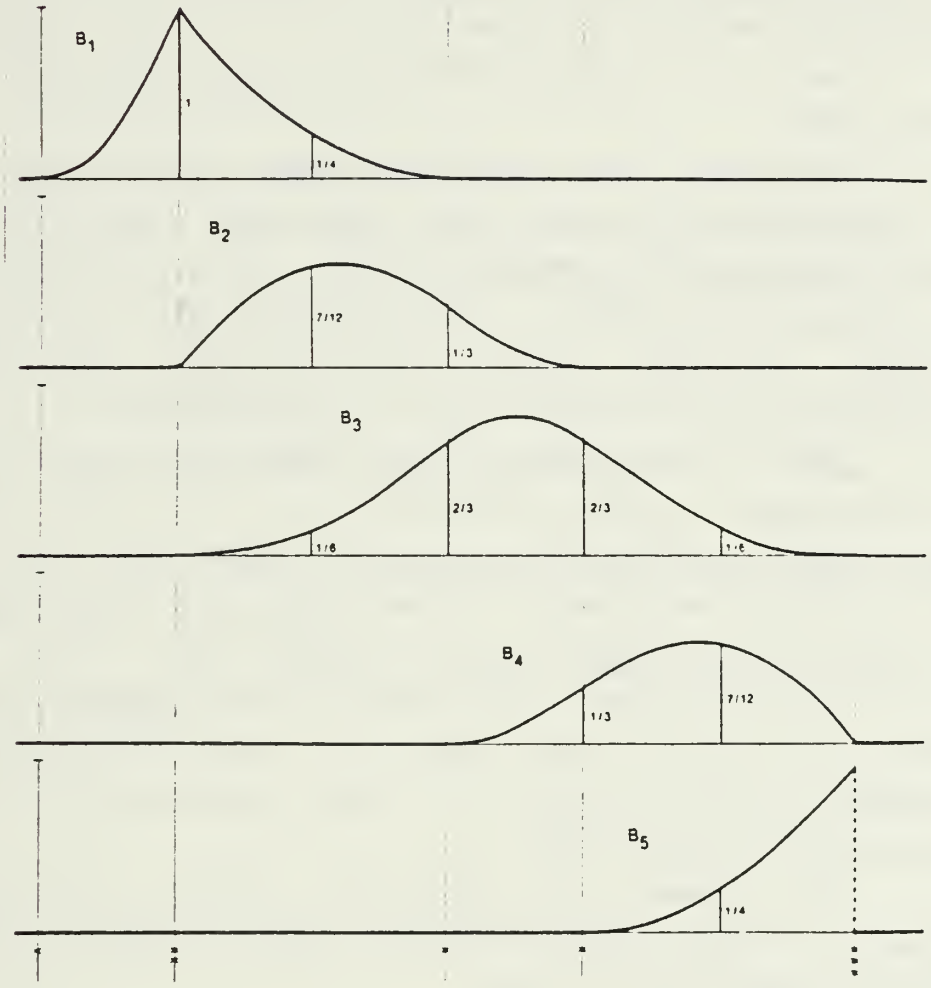

Figure 2.1 Multiple Knot Effect.

D. B-SPLINE FUNCTIONS

1. Motivations

So far, we have defined and characterized the piecewise polynomial spline as a smooth function that fits at the break points satisfying some ending point requirements.

There are some drawback associated with those constraints:

(i) If we are processing experimental data points, most of the time we are not interested to fit a curve exactly through this set of points, for the simple reason that there is an intrinsic error associated with those measurements. We rather prefer to fit a curve that presents a tradeoff between closeness of fit and smoothness.

(ii) If the number of data points is large, although the tridiagonal system associated with the spline function can be solved by efficient algorithms [Ref. 7], it is still time consuming and storate expensive. The image processing work is a clear example of this limitation.

(iii) The lack of ability to manipulate the curve fitting, so that we can not alter local behavior without altering the whole curve, is a bad characteristic of many curve fitting algorithms.

The concept of function spline representation through a linear combination of B-spline basis was introduced by Schoenberg in 1967.

The mathematical framework of the B-spline basis is the linear space theory. Here we will deal with the applications and properties of these basis. For more information about these fundamentals see [Ref. 10].

One of the first and probably the most important application of B-spline for image processing was carried out by Andrew, [Ref. 4]. In Andrew's work, the image was sectioned into subimages, each one with a different uniform

knot density related to the pixels distribution over the image, and a Bi-cubic B-spline function based on these knots was used to represent the image.

In Chapter III an algorithm based on B-spline to do automatic knots selection from the image contours is presented.

2. B-Spline Definition

As we have seen before, we are concerned with the family of splines with knots of the multiplicity one. This means that the set of given break points coincides with the set of knots. In other words, we are interested in the maximum smooth spline function, i.e., that with continuity up to the K-1 derivative, where K is the order of the spline. We will be using along this work K = 3, i.e., cubic spline.

For this class of functions, we define the B-spline representation as follow:

Given a set of knots $(t_i)_0^n$ and 2(k+1) additional knots satisfying

$$t_{-k} \leq t_{-k+1} \leq \ldots \leq t_0 = a \qquad (2.17)$$

$$b = t_{n+1} \leq t_{n+2} \leq \ldots \leq t_{n+k+1}$$

but which is further arbitrary.

Any spline function S(x) of degree K with continuous derivative up to K-1, can be represented by

$$S(x) = \sum_{i=k}^{n} c_i M_{i,k+1}(x) \qquad (2.18)$$

where $M_{i,k-1}(x)$ is the normalized B-spline basis and $(c_i)_{-k}^{n}$ is the coefficient sequence which uniquely represents the function $S(x)$ with respect to the B-spline basis $(M_{i,k+1}(x))$.

The B-spline function is constructed from certain linear combination of truncated power functions, so that each basis vanishes outside a small interval.

Among several definitions for the B-spline (see [Ref. 3]), we are going to select the one based on the divided difference of the truncated power function.

Given a sequence of knots $(t_i)_0^n$ the i^{th} B-spline basis of order m (i.e., degree K) is defined by

$$N_{i,k+1,t}(x) = (t_{i,k+1} - t_i)\, \Delta_t^{k+1}(t_i \ldots t_{i+k+1})\, g_k(t,x) \qquad (2.19)$$

where

$$g_k(t,x) = (t-x)_+^k \quad \begin{cases} (t-x)^k & \text{if } t \geq x \\ 0 & \text{if } t < 0 \end{cases}$$

and

$\Delta_t^{\ell}(z_i \ldots z_{i+\ell})f(t)$

stands for the ℓ^{th} divided difference of the function $f(t)$ at the points z_i, z_{i+1}, ... $z_{i+\ell}$. Note that the divided difference of $(t-x)_+^k$ is obtained by fixing x and considering $(t-x)_+^k$ as a function of t alone.

The resulting number depends, of course, on the particular value of x we choose, i.e., the resulting number varies as we vary x and so we obtain eventually the function $N_{i,k+1,t}(x)$.

In order to simplify the notation, we will use N_i instead of $N_{i,k+1,t}(x)$ as long as the K and the t can be inferred from the context.

The normalized B-spline is defined as

$$M_i = ((K+1) / (t_{i+k+1} - t_i) N_i \qquad (2.20)$$

3. Properties of B-Spline Functions

The B-spline basis enjoy very nice properties which make them quite superior compared with another proposed basis. As we will see later on, some of these properties are very useful for image processing purposes.

(i) Local basis

It follows that only K+1 B-splines have no zero value at any particular interval $[t_i, t_{i+1}]$, i.e.,

$$M_{i,k+1}(x) = 0 \qquad \text{if } x \leq t_i \text{ or } x > t_{i+k+1} \qquad (2.21)$$

This property suits very well the nonglobal characteristics of the pixels in a image plane; besides, at most k+1 (where K = 3 for cubic spline), basis are required to evaluate the function S(x) in any point between two generic knots $[x_j, x_{j+1}]$

$$S(x) = \sum_{i=j-k}^{j} c_i M_{i,k+1}(x) \qquad (2.22)$$

(ii) Non-negative basis

The B-spline basis is non-negative, i.e.,

$$M_{i,k+1}(x) > 0 \text{ for } t_i < x < t_{i,k+1} \qquad (2.23)$$

Again this property is nicely related to image processing application once the signals represented are usually light intensity and are always positive quantities.

(iii) Partition of the unity

The summation of all normalized B-spline basis is equal to the unit, i.e.,

$$\sum_i M_{i,k+1}(x) = 1 \qquad (2.24)$$

(iv) Differentiation and integration

As we will see in the next subsection, the B-spline can be evaluated from a recursive relation, equation 2.25, which allows the numeric differentiation and the integration to be carried out efficiently by just differing or summation of the coefficients respectively.

For the derivative of a spline we have

$$D_j\left[\sum_i^n c_i M_{i,k+1}(t)\right] = \sum_i^n c_i^{(j+1)} M_{i,k+1-j}(t) \qquad (2.25)$$

where

$$c_i^{(j+1)} \quad \begin{cases} (k+1-j) \dfrac{(c_i^{(j)} - c_{i-1}^{(j)})}{(t_{i+k+1-j} - t_i)} & \text{for } j > 0 \\ c & \text{for } j = 0 \end{cases}$$

For the integration of a spline we have

$$\int_{x_1}^{t} \sum_{i=0}^{n} c_i M_{i,k+1}(l)\, dl = \sum_{i=1}^{n} c_i^{(-1)} M_{i,k+2}(t) \qquad (2.26)$$

where

$$c_i^{(-1)} = \sum_{j=1}^{i} \frac{c_j (x_{k+1+j} - x_j)}{k+1} \qquad i = 1,\ldots,n$$

Differentiation and integration play an important role in the image processing applications, such as edge extraction through gradient operator, removal of blur caused by linear motion and smoothing.

4. Recurrence Relation for B-Spline

Although the definition for the B-spline basis as a divided difference according to equation 2.19 is mathematically perfect, it may cause problems like loss of value significance during the numerical computation of various difference coefficients.

A numerical stable formula can be obtained by applying the Leibnitz's formula for the K^{th} divided difference of a product to the particular product.

$$(t-x)_+^k = (t-x)\,(t-x)_+^{k+1} \tag{2.27}$$

The development of the equation 2.19 can be seen in [Ref. 10]. The final result is the following recurrence relation

$$N_{i,k+1}(x) = \frac{x-t_i}{t_{i+k}-t_i} N_{i,k}(x) + \frac{t_{i+k-1}-x}{t_{i+k-1}-t_{i+1}} N_{i+1,k}(x) \tag{2.28}$$

and

$$N_{i,1}(x) \begin{cases} 1 & t_i \le x \le t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

In the following chapter, the B-spline basis will be used as fundamental entities for a data compression model.

III. APPLICATION OF B-SPLINE IN THE KNOT SELECTION

A. INTRODUCTION

As was mentioned in Chapter I, our objective is to achieve data compression.

In the first step, the image contours are generated from the data over rectangular and uniform grid representing the image. The total number of contour lines is a function of the infomration contents of the image signal and the number of contour levels chosen (layers). This last parameter is under the user's control, and we will see that in the sensitivity analysis performed during the testing phase.

The rationale to use contours in the representation of image signal is that the contours representation is a powerful technique to map three- dimensional data. As discussed in Chapter I, a monochrome images can be represented as function of two variables and seen as a surface in the space.

Contouring techniques are well known in the past, and there are several techniques to generate contours. Another important point about contour representation is that it is a quite efficient way to eliminate redundant information. Therefore, it is a suitable tool when we want to achieve image data compression.

In the second step, from the image contours generated in the first step some "important" knots are selected so that the image is represented as data over a set of uneven knots. It contains much less elements than the original data set, but preserves the important features of the original signal. In another words, our image representation with this new set requires much less bulk storage than the original data over uniformly sampled grid.

The algorithm to select the "important" knots along each contour is the main subject of this chapter. The mathematical tools introduced in Chapter II (spline and B-spline) are the basis for the algorithm developed here.

The requirements for this algorithm are stated as follow: the selection of the knots should be such that the image could be reconstructed, and that the error from the original data should be minimized.

When we deal with images represented in contours, we have two degrees of freedom: the first one is to represent each contour (plane curve), and the second one is to represent the three-dimensional effect, i.e., the relation between contours at adjacent layers.

In the next section, the particular contouring problem is studied and adapted to the image processing environment. The following sections deal with the problem of selecting "important" knots of each contour. These points are important because they provide contribution to data compression in representing the image. Informally, their density is proportional to the gradient of gray scale, i.e., in regions with abrupt changes in gray scale the density of "important" points will be high; the opposite is true for dull regions. The formal criterion for choosing "important" points is discussed in Section C of this chapter.

B. CONTOUR GENERATION

As we have seen in Chapter I, a digital image can be represented as a function of two variables. These variables are the spatial coordinates and the value of the function of each coordinate pair is the gray level (quantized) at that point.

A contour algorithm determines the sequence of points of the image function in a plane, which may be used to draw

contour lines representing equal elevation of the surface. These plane curves through which the image surface is cut perpendicularly to the gray level axis are called contour levels or simply layers.

In digital image the gray level is quantized, i.e., countable. The criterion to select the number of contour levels and the space between them is in some sense arbitrary. In this work, as we are looking for a more robust and general approach, the selection method of a number of layers and how they are located is not a function of the particular set of data processed. One of the subjects of this work is to study how sensitive is the compression ratio, the quality of the reconstructed image, and the computing time with respect to the parameters of the contouring process.

Contour generating algorithm may be constructed basically according to two different methods:

(i) The tracing method, where the contours are followed from the same starting point until they either close or intersect a boundary. The advantage of this method is that labeling is relatively easy and that the pen of a incremental plotter (when we are drawing the contour) does not move about randomly.

(ii) The other classical method is performed by dividing the image array in small subimages called cells. The size of the cell is arbitrary, and so is the number of them. The algorithm is carried out by determining all contours inside each cell independently of the other, i.e., each cell can be completely processed independent of the rest of the array of data. Although in this method the bookkeeping is more complex than in the tracing method, it has some advantages like less memory requirement, and the capability to generate contour over a much larger array.

Perhaps, the most important feature of this scheme is its hability to support parallel processing while the contour tracing method is inherently a sequential approach.

1. Contour Algorithm Selection

Based upon a survey of contouring algorithms [Ref. 1], [Ref. 5], [Ref. 9], our requirements, and our programming environment (see Appendix A), the "CONTUR" algorithm was selected from the NONIMSL Library. This algorithm is a contour following type. The algorithm generates a contour representationon which one or more (usually more) contour levels are extracted, i.e., given a matrix of numerical values of z = f(x,y), the CONTUR finds the loci of specified discrete values of z and saves these loci (contour lines) in a file with format specified by the user.

2. Contour Algorithm Modification

Our task is to extract "important" knots (x and y pairs of spatial coordinates) from the contours. In order to do so, we need to have the contours stored in a convenient way so that our algorithm, which does the selection, has access to these set of data. Other information which are necessary are:

(i) The density of contours in the neighborhood of each break point gives us indication of how the signal energy changes in that neighborhood; therefore, how "important" is that knot.

(ii) The gray value of each contour level (layer) and the number of contours that belongs to each layer. This information is helpful in the image reconstruction process.

From imput data we have:

(i) The uniform grid representing the image.

(ii) The user's controlled parameters are: 1) "NC" that is the number of levels in which the image is chopped. 2) "THRES" that is the threshold that eliminates contours with less points than "THRES". This last parameter has a smoothing effect that will be helpful when we will deal with noisy images.

The output of the contouring algorithm are:

(i) Contour record: It is a sequence of pairs (x,y) representing the coordinates of each coutour. The contours are concatenated together and there are two labels at the beginning of each one giving the number of points in the following contour and its type (opened or closed).

The Figure 3.1 shows how these sequences are organized. An end mark is used at the end of the sequence.

(ii) Character matrix: It is a matrix with the same dimension as the given image containing the projection of all contours of the image surface. The contours in this matrix are fully connected with connectivity four definition. Each layer is coded by a character starting from character A (lower contour level) through character Z (higher contour level) allowing up to 23 layers. The blank character is utilized for the background. In this manner, we have the projection of all contours of the image surface in the floor plane. This set of data turns out to be very useful when we incorporate the three-dimensional effects through the density of contours around each pair of coordinates of the contour lines.

The Figure 3.2 gives an example of a typical character matrix.

(iii) Layer record: It is similar to the contour mapping, as far as structure is concerned. The information contained here is about the gray level of each layer and the number of contours that belongs to the respective layer.

The Figure 3.3 illustrates a typical layer mapping with ten layers.

More information about how these set of data are created, stored, and used are given in Appendix A.

23	-2	←HEADER OF A CLOSED CONTOUR WITH 23 (X,Y) PAIRS
3	1	
5	1	
7	0	
8	2	
9	3	
10	4	
11	3	
13	4	
14	8	
15	9	
16	11	
15	12	
14	15	
11	16	
10	16	
9	15	
7	-1	←HEADER OF AN OPEN CONTOUR WITH SIX (X,Y) PAIRS
0	22	
1	23	
2	24	
4	23	
5	23	
5	24	
6	25	
-3	-3	←END OF THE RECORD INDICATOR

Figure 3.1 Contour Record Format.

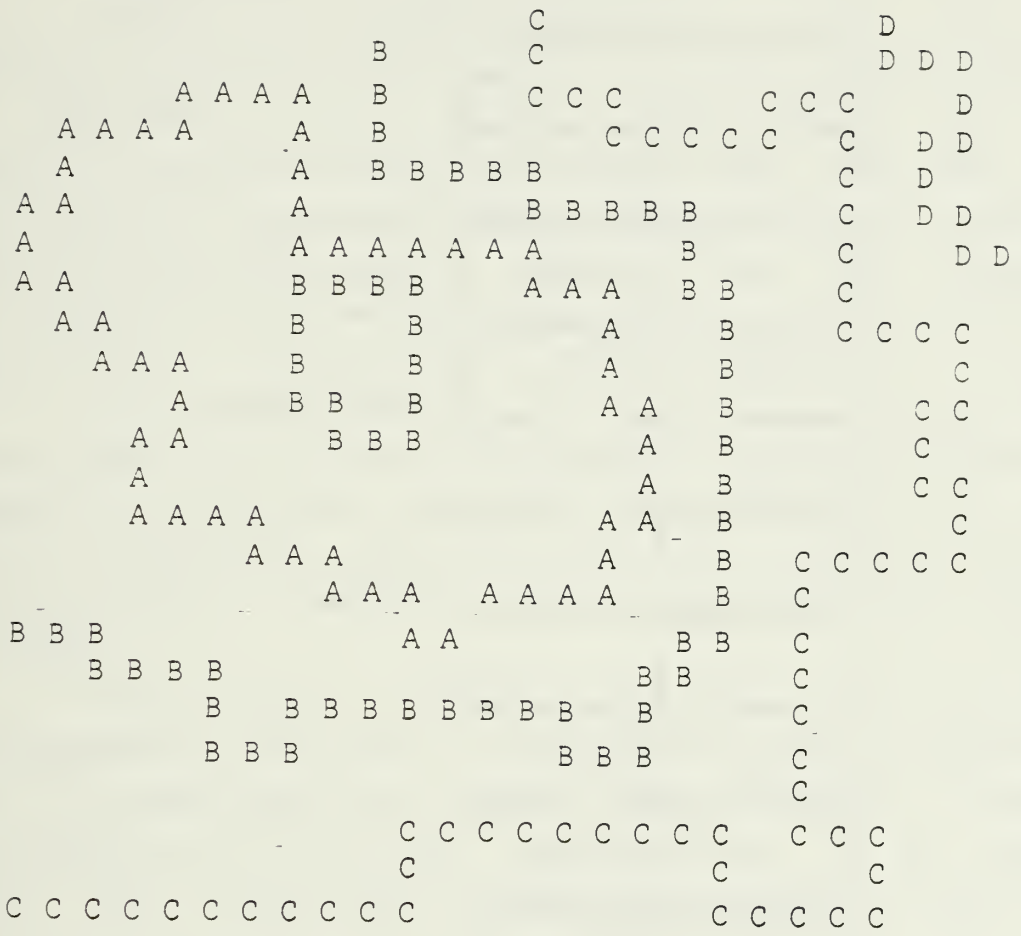

Figure 3.2 Character Matrix Format.

NUMBER OF CONTOURS IN EACH GRAY LEVEL	GRAY LEVEL VALUE	
0	45	
7	60	
18	85	
26	100	
30	115	
44	130	
42	145	
33	160	
28	185	
15	200	
8	215	
1	230	
-3	-3	← END OF THE RECORD INDICATOR

Figure 3.3 Layer Record Format.

C. KNOT SELECTION

1. Introduction

Given the contour information of an image as discussed in the previous section, we are ready to present an algorithm to fit a smooth spline function to each contour. The idea behind this algorithm [Ref. 10], [Ref. 11] is to define a measurement for the smoothness of the approximating splines and another measurement for their closeness of fitting so that these two properties, usually contradictory, could be controlled by a single parameter S.

The greatest advantage of this method compared with others [Ref. 12] is that it allows us to automate the determination of the knots. As it will be shown later on, the number of selected knots is much lower than the initial contour points; therefore, high compression rate can be achieved.

For the sake of better understanding, the algorithm is presented in two parts. In the first part, the development is restricted to a single curve (contour) without considering the effects of its neighbors. In the second part, that neighboring (three-dimensional) effect is included as an extension of the former algorithm.

2. Bi-Dimensional Case

a. Problem formulation

Given the set of data points (x_i, y_i), $i = 1,\ldots,m$ where x_i is in the range $[a,b]$ and y_i is the value of the function at the point x_i, and the positive weighting values w_i, $i = 1,2\ldots,m$, we want to determine a spline function $S(x)$ of degree K with knots t_j, $j = K+1, K+2 \ldots m-K$, such that we have a tradeoff between the following goals:

(i) The prescribed value y_i should fit close enough.

(ii) The approximating spline should be smooth enough, in the sense that the discontinuity of its K^{th} derivative is as small as possible.

Mathematically, this criterion can be stated in the following way:

(i) For a measurement of closeness of fit given by

$$\rho(\bar{c}) = \sum_{j=1}^{m} w_j (y_j - S(x_j))^2 \tag{3.1}$$

where

$$S(x_j) = \sum_{i=-k}^{n} c_i N_{i,n+k}(x_j) \tag{3.2}$$

is the representation of a function through the normalized B-spline basis, for n selected knots.

(ii) For the smoothing norm as a measurement of the lack of smoothness given by

$$n(\bar{c}) = \sum_{r=1}^{n} (d_r)^2 \tag{3.3}$$

where d_r represents the discontinuity jump of the $s^{(k)}(x)$ at t_r, i.e.,

$$d_r = S^{(k)}(t_r + 0) - S^{(k)}(t_r - 0) \tag{3.4}$$

or

$$d_r = \sum_{i=k}^{n} a_{i,r} c_i \qquad (3.5)$$

Then we can state our approximation criterion as follow:

minimize

$$n(\bar{c}) \qquad (3.6)$$

under the constraint

$$\rho(\bar{c}) \leq S \qquad (3.7)$$

where S is a nonnegative parameter under the user's control called smoothing factor.

b. Problem solution

As we can see, our task is to minimize a function under a constraint. The Lagrange method is quite suitable for this kind of problem, and it is used here. From equations 3.6 and 3.7 we can state immediately that

$$\text{min}k(p,z,\bar{c}) = \sum_{r=k+2}^{n-k-1} d_r^2 + p\left[\sum_{i=1}^{m} w_i (y_i - S(x_i))^2 + z^2 - S\right] \qquad (3.8)$$

From the conditions $\frac{\partial k}{\partial p} = 0$, $\frac{\partial k}{\partial z} = 0$ we get

$$\rho(\bar{c}) = S - z^2 \qquad (3.9)$$

$$p.z = 0 \qquad (3.10)$$

As we are interested in the nontrivial case, i.e. $p \neq 0$ from equation 3.10 we have $z = 0$. It can be shown [Ref. 13] that with every positive value of p there is a single spline $S_p(x)$ which minimizes

$$K(p,o,\bar{c}) = \sum_{r=k+2}^{n-k+1} d_r^2 + p\left[\sum_{i=1}^{m} w_i (S_p(x_i) - y_i)^2 - S\right] \qquad (3.11)$$

Let's define the function $F_n(p)$ such that

$$F_n(p) = \sum_{i=1}^{m} w_i (S_p(x_i) - y_i)^2 \qquad (3.12)$$

From equation 3.9 and our assumption on nontrivial case, we have that

$$F_n(p) = S \qquad (3.13)$$

Therefore, our problem ends up in solving the simultaneous equations 3.11 and 3.12. For the first one, we evaluate the B-spline coefficients by minimizing the function $K(p,o,\bar{c})$ for a fixed and positive value of p. For the second one, we find a positive root of $F_n(p) = S$.

Leading equation 3.5 into equation 3.11 and applying the necessary condition for minimum, i.e., $\frac{\partial k}{\partial c_\ell} = 0$, it turns out that

$$\sum_{j=1}^{n-k+1} c_j \sum_{r=k+2}^{n-k+1} a_{j,r}\, a_{\ell,r} + p \sum_{j=1}^{n-k+1} c_j \left[\sum_{i=1}^{m} w_i N_{i,k}(x_i) N_{\ell,k}(x)\right] = \qquad (3.14)$$

$$= p \sum_{i=1}^{m} w_i y_i N_{\ell,k}(x_i) \qquad \ell = 1,2,\ldots,n-k-1$$

Rewriting equation 3.14 in matrix form we have

$$GC = Z \tag{3.15}$$

where

$$G = A + p^1 B$$

$$B_{s,j} = \sum_{r=k+2}^{n-k+1} a_{s,r} a_{j,r}$$

$$A_{s,j} = \sum_{i=1}^{n} w_i N_{s,k}(x_i) N_{j,k}(x_i)$$

$$Z_i = \sum_{i=1}^{n} w_i y_i N_{i,k}(x_i)$$

$$C^J = (C_1, C_2, \ldots, C_{n-k-1})$$

Due to the properties of the locality of the B-splines and the assumption that $p > 0$, the above matrices A, B, and G are positive, semidefinite, and banded so that the equation 3.15 can be solved easily. For more details see Martin [Ref. 14].

For the solution of the equation 3.12, an interactive process used starting from the boundary solutions of equation 3.15, which approaches to the least square spline $S_n(x)$ as p approaches to infinity, and to the least square polynomial $P_k(x)$, as p approaches to zero.

Mathematically, it can be expressed in this way

$$F_n(\infty) = \sum_{i=1}^{n} w_i (y_i - S_n(x_i))^2 \tag{3.16}$$

$$F_n(0) = \sum_{i=1}^{n} w_i (y_i - p_k(x_j))^2 \qquad (3.17)$$

It can be proved that F(p) is continuous, monotonically decreasing, and convex for $p > 0$ [Ref. 13]. Therefore, there is a unique root $\bar{p}$ of $F(p) = S$. In order to determinate $\bar{p}$, any rootfinder method can be employed.

The following rational interpolation method is chosen due to its fast convergence and simplicity.

We start with three values of p

$$P_1 = 0, \quad P_2 = -(F_n(0) - S)/F_n(\infty) - S, \quad P_3 = \infty \qquad (3.18)$$

and the corresponding values of F(p) (F_1, F_2, F_3) are used to calculate the parameters u, v, w of an auxiliary function of the form

$$R(p) = up + v / p + w \qquad (3.19)$$

By setting $R(p) = S$, a new approximation p^* for $\bar{p}$ is found.

$$p^* = p_1(F_1 - F_2)(F_3 - S) - P_2(F_1 - F_3)F_1 - S)/(F_1 - F_2)(F_3 - S) \qquad (3.20)$$

if $p_3 = \infty$

$$p^* = \frac{-(F_3 - S)(F_1 - F_2)p_1 p_2 + (F_2 - S)(F_3 - F_1)p_3 p_1 + (F_1 - S)(F_2 - F_3)p_2 p_3}{(F_3 - S)(F_1 - F_2)p_3 + (F_2 - S)(F_3 - F_1)p_2 + (F_1 - S)(F_2 - F_3)p_1}$$

if $p_3 \neq \infty$

The value of P_1 , P_2, P_3 are adjusted according to the following scheme if

$$F_2 < S \; ; \; P_1 \leftarrow P_2 \quad P_2 \leftarrow p^* \tag{3.21}$$

$$F_2 > S \; ; \; P_3 \leftarrow P_2 \quad P_2 \leftarrow p^*$$

so that the interval (p_1, p_2) is permanently reduced, while

$$p_1 < p_2 < p_3 \; ; \; F_1 > S \; ; \; F_3 < S \tag{3.22}$$

The criterion to stop the interaction is chosen as

$$|F(p^*)-S| \; / \; S \quad \leq \varepsilon \tag{3.23}$$

Because we are dealing with a smooth process, it does not make sense to have ε too small; a typical value of 0.001 is suggested by Dierckx [Ref. 12].

c. Parametric splines

Because our contours are arbitrary plane curves opened or closed, the condition $x_j < x_{j+1}$ is not necessarily fulfilled. Consequently, an adjustment should be done in order for the theory presented so far be applicable to our contouring problem. The idea to associate each pair of break points (x_j, y_j) with a monotonic increasing parameter z_j ,

$j = 1,2...,m$ appears to be quite reasonable. As it was mentioned in Section A of this chapter, z_j is the free parameter and x_j and y_j are the dependent ones.

Then, in this new parametric form, we can write the representation of the x and y coordinates using the B-spline approach.

$$S_x(z) = \sum_{i=-k}^{n} c_{x,i} M_{i,k+1}(z) \qquad (3.24)$$

$$S_y(z) = \sum_{i=-k}^{n} c_{y,i} M_{i,k+1}(z) \qquad (3.25)$$

where

$x = S_x(z)$

$y = S_y(z) \qquad a \le z \le b$

are the parametric representation of a contour.

Analogous to the former case, the minimization of a parametric curve can be stated as:

<u>minimize</u>:

$$\sum_{j=1}^{n} [S_x^{(k)}(t_j+0) - S_x^{(k)}(t_j-0)]^2 + [S_y^{(k)}(t_j+0) - S_y^{(k)}(t_j-0)]^2 \qquad (3.26)$$

<u>subject to the constraint</u>:

$$\sum_{j=1}^{m} w_j [(x_j - S_x(z_j))^2 + (y_j - S_y(z_j))^2] \le S \qquad (3.27)$$

Then, analogously to the minimization problem 3.8, we end up with the B-spline coefficients $c_{x,y}$ and $c_{y,i}$ of the two splines $S_{xp}(z)$ and $S_{yp}(z)$, derived for positive values of p, as the least square solution of the systems:

$$\sqrt{w_j}\sum_{i=k}^{n} c_{x,i} M_{i,k+1}(z_j) = \sqrt{w_j}\, x_j \qquad j = 1,2,\ldots,m \tag{3.28}$$

$$\frac{1}{\sqrt{p}}\sum_{i=k}^{n} c_{x,i} a_{i,j} = 0 \qquad j = 1,2,\ldots,n$$

$$\sqrt{w_j}\sum_{i=k}^{n} c_{y,i} M_{i,k+1}(z_j) = \sqrt{w_j}\, y_j \qquad j = 1,2,\ldots,m \tag{3.29}$$

$$\frac{1}{\sqrt{p}}\sum_{i=k}^{n} c_{y,i} a_{i,j} = 0 \qquad j = 1,2,\ldots,n$$

when p is given as the value of the positive root of $F(p) = S$, now with

$$F(p) = \sum_{j=1}^{m} w_j \left[(x_j - S_{x,p}(z_j))^2 + (y_j - S_{y,p}(z_j))^2 \right] \tag{3.30}$$

About the parameter z, Adams [Ref. 16] suggests two methods, and Dierckx [Ref. 15] gives four choices. As we are looking for a general case, because we do not know a priori the shape of these contours, we choose the Dierckx's first proposal as follows:

$$z_1 = 0 \tag{3.31}$$

$$z_j = z_{j-1} + d_j \qquad j = 1,2,\ldots,m$$
$$a = z_1 \; ; \; b = z_m$$

where

$$d_j = \sqrt{(\Delta x_j)^2 + (\Delta y_j)^2} \tag{3.32}$$

and

$$\Delta x_j = x_j - x_{j-1} \qquad (3.33)$$

$$\Delta y_j = y_j - y_{j-1}$$

Another important observation should be made about the parametric spline as far as smoothing of contours is concerned. As we have seen, the contour may be opened or closed curves. For opened contours the parametric form applies directly, while for the closed ones a small modification should be done. We just add the periodic condition: $S^{(j)}(a) = S^{(j)}(b)$ $j = 1,\ldots,K$ as a constraint to the system formed by equations 3.28 and 3.29.

3. Three-Dimensional Case

a. Introduction

So far, we have seen the application of the function spline to fit a smooth parametric curve through a set of m breaking points by generating a set of n knots, which is much lower than m.

The number of knots added in each iteraction depends on two factors: 1) The number of knots added at the last time. 2) The reduction in the weighted sum of the squared residuals $F_{n_j}(\infty) - F_{n_{j-1}}(\infty)$, which is always positive by the fact that the knots are only added and not relocated.

For our purpose, besides the automatic knot selection, another important feature of locating knots is that it tends to accumulate more knots in the regions where the underlying data is difficult to approximate than in areas where a smooth behavior is present.

b. Algorithm extension

The fact that the regions with greater fitting error are granted with more knots, guides us to a reasonable approach to introduce the three-dimensional effects. We do this by assigning to each break point a weight that is function of the density of the contour in its neighborhood.

Then, for each pair of coordinates (x,y) of every contour, we estimate the density of contour lines around it and assign a weight to that point in this way. If the density is high, the weight will be high and the opposite is true when the density is low.

The Figure 3.4 shows an heuristic scheme to estimate the density of contour lines at any point on a particular contour. The strategy works in the following way: centered in each contour's point (x,y), we count in four orthogonal directions (E, N, W, S) the number of occurrencies of contours at different layers, i.e., N_E , N_N , N_W , N_S and select the highest counting among the four directions as the representative of the density of the contours, i.e., $N = MAX\ (N_E , N_N , N_W , N_S)$.

The span in each direction is a parameter "DISP" that is under the user's control. Then, for every pair, the weight is assigned according to the following formula:

$$W_i = W_0 + N_i \Delta W \qquad (3.34)$$

where

W_i : actual weight

W_0 : reference weight $\quad i = 1,2,\ldots,m$

N_i : maximum number of contours crossing the current area

ΔW : incremental weight

It is the subject of this work to measure and reveal the influency of the above parameters in this image compression method and the reconstructed image.

In order to clarify the concept of the density measurement, let's show an example. In Figure 3.4 we have the superimposition of the contour record, Figure 3.1, and the character matrix, Figure 3.2. The character matrix is composed of fully connected contours represented in letters, while the contour record consists of the contour represented by uneven spaced knots marked with a circle. The contour density measurement is accomplished by moving the isometric cross along each contour so that it is centered in every contour record point (circled character in Figure 3.4), and an information about distribution of contours in that specific neighborhood can be valiated by checking four directions spaced 90 degrees apart. The length of each arm of the isometric cross is set by the parameter DISP under the user's control. As we will see in the testing phase, DISP equals five is a good choice.

For the particular location of the cross in the referred Figure 3.4 we have the following measurements and computations: $N_E = 2$ $N_N = 3$ $N_W = 1$ $N_S = 0$; therefore, $N = 3$ assuming $W_0 = 1$ and $\Delta W = 0.2$, the weight for the break point (x_i, y_i) is $W_i = 1.6$.

Another parameter that has a direct affect in the selection of the knots is the smoothing factor, namely S, defined in equation 3.13. This parameter works like a tuning element that controls the fitness and the smoothness. As we decrease the factor S, the function fits closer while it becomes less smooth due to the large number of selected knots. The opposite is true when we increase S.

There is no optimum value for S in a wide sense. It will depend on how close or smooth we want to fit, consequently how much data compression and fidelity we want to tradeoff.

The factor S is an increasing function of the initial number of break points m. In our testing phase it was realized that the computing time increases faster for values of m of the order of hundreds. The following expression for S in terms of m has demonstrated to be a good compromise between computing time and amount of data compression.

$$S = \sqrt{m}\ (m \text{ MOD } 100+1) \tag{3.35}$$

where m MOD 100 means: m modulo 100.

In the next chapter, the inverse process will be followed. A uniform grid will be constructed from the compressed set of knots and compared with the original image.

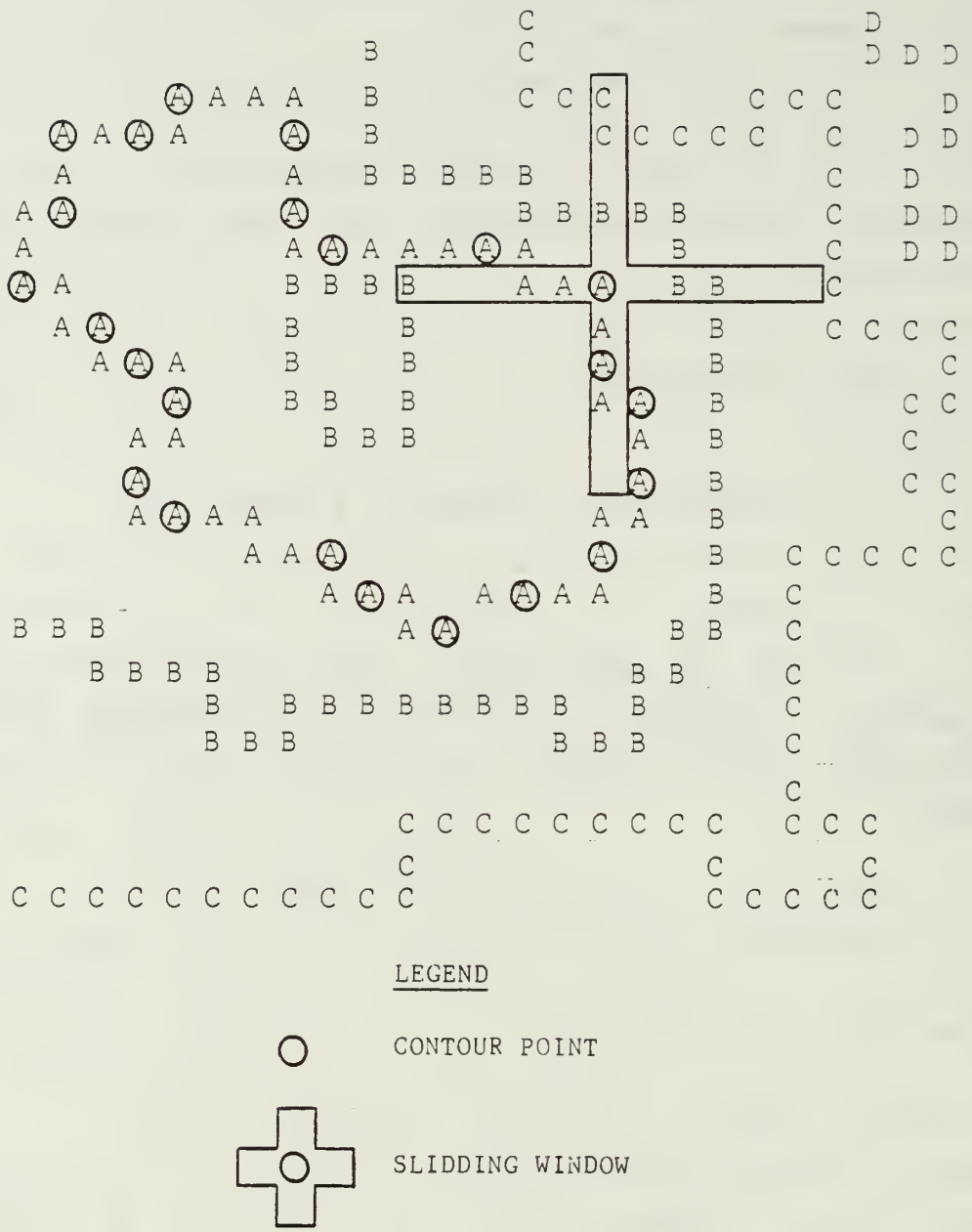

Figure 3.4 Pictoric View of the Density Measurement Scheme.

IV. IMAGE RECONSTRUCTION

A. INTRODUCTION

In Chapter III the development of an algorithm to compress the digital image was shown. Now, we are interested in reconstructing the compressed image and compare it with the original one. In other words, we want to have a fidelity criterion to measure the quality of the output.

From the former chapter we have the following scenario: a set of knots unevenly spaced lying along the image contours. The density of knots is higher in busy regions (where faster changes in gray level occur) then in doll areas. In order to reconstruct the image over (uniform grid), it requires an algorithm that can generate a surface from a scattered data points.

A variety of algorithmic ideas has been developed for this kind of problem or closely related problems. A survey was done in this area, [Ref. 17], [Ref. 18], [Ref. 19], [Ref. 20], [Ref. 21], [Ref. 22], and two of the referred methods based on different approaches were chosen.

Before going to the description of the select methods and reconstruction process, it is necessary to introduce the fidelity criterion measurement.

B. RECONSTRUCTION FIDELITY CRITERIA

In image processing system the output image sometimes is the final product to be viewed by people or it is the input to another image processing unit. The concern is that the human viewing mechanism has quite peculiar characteristics, [Ref. 1]. Thus, what is suitable for a person may not be (and usually is not) suitable for machine and vice-versa.

Therefore, it is reasonable to define two fidelity criteria: the first one is based on the human judgement and it is called subjective (qualitative) fidelity criteria. The second one is based on mathematical relationship and it is called objective (quantitative) criteria. In our choice of the objective fidelity criteria, we will look for those which resemble the human subjective viewing more.

1. Objective (Quantitative) Fidelity Criteria

In a image processing system, like Figure 1.1, an accepted definition for pixel error is given by the difference between an input pixel gray level and the corresponding output pixel gray level. Formally, we say that for a given input image $f(x,y)$, as defined in Chapter I, the error $e(x,y)$ at each pixel is given by

$$e(x,y) = f(x,y) - g(x,y) \tag{4.1}$$

where $g(x,y)$ is the output image signal; (x,y) are the spatial coordinates, $(x,y = 0,1,\ldots,n-1)$.

Based on this elemental definition of pixel error $e(x,y)$, several definitions of the error which express the overall quality of the image arise, and some of them are listed below:

(i) Root mean square error

It is the most popular error measurements definition, not only in image processing, but in many other branches of the engineering science. It is defined as the square root of the squared error averaged over the image array.

$$e_{rms} = [\langle \bar{e}^2 \rangle]^{1/2} \tag{4.2}$$

where

$$\langle \bar{e}^2 \rangle = \frac{1}{N^2} \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} e^2(x,y)$$

N^2 is the total number of knots.

The averaging sense of this definition resembles that of the intrinsic human visual system; it tends to filter out high frequencies and introduce smoothing effects in the image.

(ii) Maximum deviation

This definition, as its name indicates, assigns to the error the maximum absolute value of the pixel error difference between the input and the output. It is quite often used by the people in graphical area. For image representation by intensity of gray level, it is not quite appropriated because it can give measurements disimilar to the subjective fidelity criteria, due to the filtering properties of the human eye.

$$e_{MAX} = MAX|f(x_i, y_j) - g(x_i, y_j)| \qquad (4.3)$$

$$i,j = 0,1,\ldots,n-1$$

(iii) Relative error [Ref. 4]

It is similar to the root mean square error. The difference is that the normalization is done not by the number of pixels N^2 but by the square of the input signal $f(x,y)$.

$$e_{REL} = \frac{\sum_{x=0}^{n-1} \sum_{y=0}^{n-1} [f(x,y) - g(x,y)]^2}{\sum_{x=0}^{n-1} \sum_{y=0}^{n-1} (f(x,y))^2} \qquad (4.4)$$

(iv) Root mean square of signal to noise ratio

If we interpret $e(x,y)$ as the noise introduced by the system at each pixel, we can see the output image as the input signal added with the noise $e(x,y)$. Then the root mean square of the signal to noise ratio is defined by

$$(SNR)_{RMS} = \left[\frac{\sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g^2(x,y)}{\sum_{x=0}^{n-1} \sum_{y=0}^{n-1}} \right]^{1/2} \qquad (4.5)$$

2. Subjective (Qualitative) Fidelity Criteria

Contrary to the previous section, we can not expect here mathematical criteria by the simplest fact that we are not dealing with deterministic matter, rather it is a psychovisual effect. Nevertheless, there are some characteristics of the human visual system which are general, such as, the logarithmic sensitivity to light intensity, the ability to differentiate about 30 gray levels and about 120 colors.

The integration effect is another feature of the human eye that tends to filter out the abrupt changes; that implies the introduction of a smoothing effect.

One absolute scale was developed by Panel 6 of the Television Study Organization [Ref. 26] and it is reproduced here:

(i) Excellent: The image is of extremelly high quality, as good as you could desire.

(ii) Fine: The image is of high quality providing enjoyable viewing. Interference is not objectionable.

(iii) Marginal: The image is of acceptable quality. Interference is not objectionable.

(iv) Inferior: The image is of poor quality but you could watch it. Objectionable interference is definitively present.

(v) Unusable: The image is so bad that you could not watch it.

The measurement should be done among a reasonable number of persons and the results should be averaged.

In the next sections two image reconstruction processes will be presented, namely thin plate spline and triangulation approaches.

In the next chapter, subjective and objective measurements will be performed on the output images reconstructed by the mentioned approaches.

C. THIN PLATE SPLINE METHOD

1. Background

The thin plate spline was first mentioned by Harder and Desmaires, [Ref. 23]. Franke, [Ref. 20], reported encouraging results from an algorithm for smooth interpolation of scattered data by local thin plate spline. The presentation of Franke's algorithm and its application for the reconstruction of the compressed image produced in the previous chapter is the subject of this section.

The algorithm is based on a weighted sum of locally defined thin plate splines, and the final result is a C^1 surface. Despite of the similarities of the mathematical relations and notations described here with those of Chapter III, we will follow as close as possible Franke's notation because as we have seen, it is not the unique solution for the problem, and the image reconstruction phase is functionally isolated from the previous steps.

Before introducing Franke's algorithm itself, it is convenient to define some terms and set up some notation.

Grid: It is a composite structure of elements (nodes) arranged in a two-dimensional fashion, such that rows and columns intercept orthogonalwise. A uniform square grid is a lattice with some number of rows and columns equally spaced.

Lattice: same as grid.

Node (i,j): It is the point where the i-row and the j-column cross each other.

Rectangle $R_{i,j}$: It is defined for each node (i,j) not belonging to the border of the grid. It comprises the region inside the polygon defined by the four vertices (i-1, j-1), (i+1, j-1), (i-1, j+1), (i+1, j+1).

Subrectangle: It is the region defined by two adjacent rows i and i+1 crossed by two adjacent columns j and j+1. As we can see, each rectangle $R_{i,j}$ contains four subrectangles. The Figure 4.1 shows a section of a uniform grid with the rectangle $R_{i,j}$ and a subrectangle highlighted.

Let's now state the formal definition of the problem: given a set of N scattered points (x_k, y_k, f_k), K = 1,...,N, a locally defined function F(x,y) which has the property $F(x_k, y_k) = f_k$, K = 1,...,N, is to be constructed, and from this function we want to generate a uniform grid through the evaluation of the function F(x,y) in the nodes of a specified lattice. In our particular case here the grid has the number of rows "Nx" equals the number of columns "Ny", and it is Nx = Ny = 256. Therefore, F(x,y) is to be evaluated in 65536 nodes of the grid. The parameters of the function F(x,y) represent the discrete spatial coordinates, rows, and columns of the uniform square grid respectively. In our particular case, x and y are integers ranging from zero to 255.

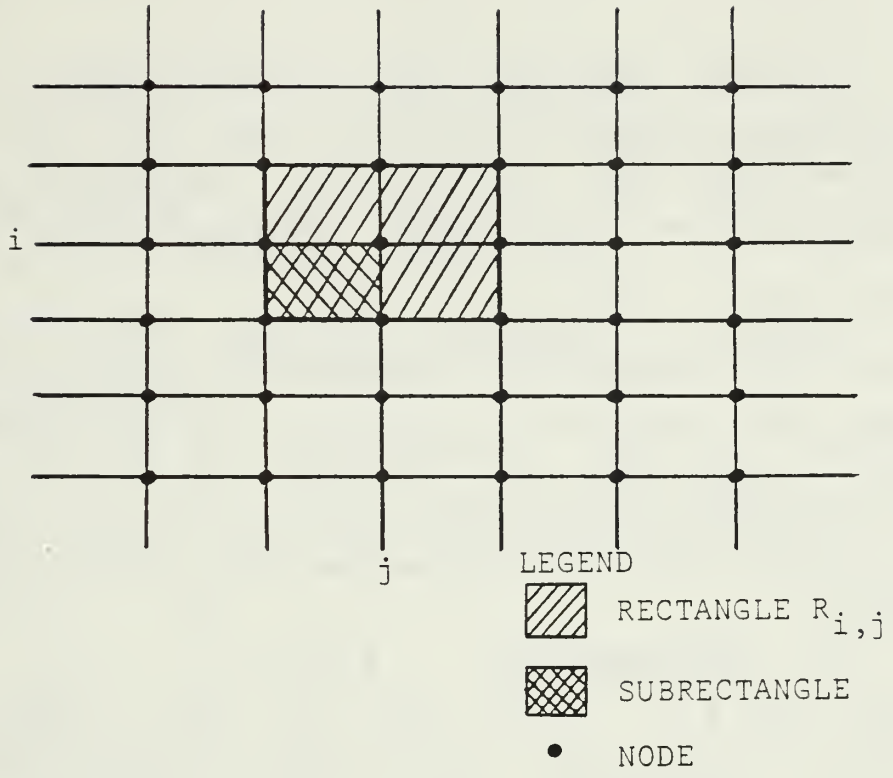

Figure 4.1 Typical Rectangle and Subrectangle in a Grid.

The overall interpolant is of the form

$$F(x,y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} W_{i,j}(x,y)\, Q_{i,j}(x,y) \qquad (4.6)$$

where $F(x,y)$ is the value of the function (gray level) at the node (x,y) defined above.

The summation is performed over the nodes of an auxiliary grid (dummy variables i,j). The details of this auxiliary lattice will be given in the next section.

$W_{i,j}(x,y)$, $i = 1,\ldots,n$, $j = 1,\ldots,n$: are the weight functions relative to the grid node (i,j) and evaluated at the point (x,y) that represent the discrete spatial coordinate of the output image (uniform square 256x256 grid).

$Q_{i,j}(x,y)$: analogous to the weight function, it is the local thin plate spline approximation relative to the node (i,j) and evaluated at the point (x,y) enjoying the property that $Q(x_k, y_k) = f_k$. We will see more on weight functions and thin plate spline later on.

The function F(x,y), equation 4.6, has the following properties:

(i) Interpolation: $F(x_k, y_k) = f_k$, $k = 1,2,\ldots,N$ where f_k is the given data point (gray level) at the point (x_k, y_k), and N is the number of given points.

(ii) Smoothness: F(x,y) is at least as smooth as the $W_{i,j}(x,y)$ and $Q_{i,j}(x,y)$; usually C^1 continuity is guaranteed for all points.

(iii) Local dependence on the data: For a fixed (x,y) the weight function does not vanish for at most four terms in the summations defined in the equation 4.6, i.e., $W_{i,j}(x,y)$ have a small support.

Similar to the B-spline discussed in Chapter III, this property is highly desirable because of its implication in reducting the processing time during the evaluation of F(x,y).

For now it is important to bear in mind the scenario that we have: a set of scattered data (x_k, y_k, f_k), $K = 1,\ldots,N$, and two uniform square grids constructed over the region containing the given points. The auxiliary grid where the thin plates are defined and weighted , and the output grid representing the reconstructed image whose nodes (x,y) are the points in which the function F(x,y), equation 4.6, is evaluated.

2. Selection of the Weight Function

According to the previous section, it was stated that the weight functions have a local support. Although there is no theoretical constraint in the shape of the support regions, in practice it is common to use rectangular regions, which simplifies enormously the problem without loss of generality.

The use of support regions which are rectangle have specific advantages in terms of controlling the number of support regions in which a particular point (x,y) lies, and of course it simplifies the determination of those regions.

From these ideas the notion of auxiliaty uniform square grid comes naturally. It is constructed over the given set of points (x_k, y_k, f_k), K = 1,...,N, and its objective is to determine the region (rectangles $R_{i,j}$) where the thin plates will be evaluated and weighted. The auxiliary grid is constructed in the following way: the user supply an estimative of the average number of points per rectangle, namely "NPPR", and the algorithm using the number of given points N and NPPR evaluates the number the number of internal rows and columns that composes the auxiliary grid, n_x and n_y respectively.

The equation 4.7 shows the described relation between the two input parameters N, NPPR and the refinement of the grid, $n_x = n_y$.

$$n_x = \text{NINT}[(4N/\text{NPPR})^{1/2} - 1] \qquad (4.7)$$

where the function NINT means closest integer.

The weight function $W_{i,j}(x,y)$ is locally defined for each internal node (i,j) and its support is the rectangle $R_{i,j}$. The auxiliary uniform square grid with n_x+2 rows and

n_y+2 columns can be represented by the positive integer sequence of rows and columns satisfying

$$\tilde{x}_0 < \tilde{x}_1 < \ldots < \tilde{x}_n < \tilde{x}_{n+1} \qquad (4.8)$$

$$\tilde{y}_0 < \tilde{y}_1 < \ldots < \tilde{y}_n < \tilde{y}_{n+1}$$

where each pair $(\tilde{x}_i, \tilde{y}_i)$ represents one node of the auxiliary grid. We call this sequence auxiliary lattice node sequence.

The tilde notation was introduced to differentiate the auxiliary grid nodes from the given points, spatial coordinates (x_k, y_k) of selected knots and the spatial coordinates of the output grid (x,y), the reconstructed image.

Let now $H_3(s) = 1-3s^2+2s^3$ be the Hermite's cubic satisfying $H_3(0) = 1$, $H_3(1) = 0$, $H_3'(0) = 0$, $H_3'(1) = 0$, as shown in Figure 4.2.

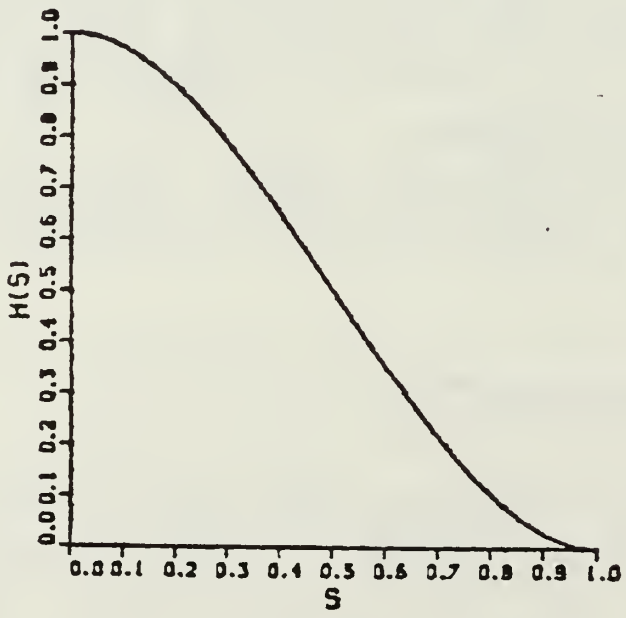

Figure 4.2 Hermite's Cubic.

Based on the Hermite's cubic polynomial and the auxiliary lattice node sequence, we define piecewise cubics with C^1 continuity, which do not vanish on two adjacent intervals, and satisfy:

(4.9)

$$V_1(x)\begin{cases}1 & , x < \tilde{x}_1 \\ H_3\left(\frac{x-\tilde{x}_1}{x_2-x_1}\right) & , \tilde{x}_1 \le x \le \tilde{x}_2 \\ 0 & , x \ge \tilde{x}_2\end{cases}$$

$$V_i(x)\begin{cases}0 & , x < \tilde{x}_{i-1} \\ 1-V_{i-1}(x) & , \tilde{x}_{i-1} \le x < \tilde{x}_i \\ H_3\left(\frac{x-x_i}{x_{i+1}-x_i}\right) & , \tilde{x}_i \le x < \tilde{x}_{i+1} \\ 0 & , x \ge \tilde{x}_{nx-1}\end{cases}$$

$$V_{nx}(x)\begin{cases}0 & , x < \tilde{x}_{nx-1} \\ 1-V_{nx-1}(x) & , \tilde{x}_{nx} \le x < \tilde{x}_{nx} \\ 1 & , x \ge \tilde{x}_{nx}\end{cases}$$

The $u_i(y)$ is dual. Then we can finally define the weight function as

$$W_{i,j}(x,y) = V_i(x)\,U_j(y) \qquad (4.10)$$

$$i = 1,\ldots,n_x$$
$$j = 1,\ldots,n_y$$

As we can see from the definition of equation 4.9, the support of the weight function except for the boundaries

of the lattice is $R_{i,j}$. The referred equation formalizes the local behavior of the scheme as was mentioned early in this section.

3. Thin Plate Spline Basis

The theoretical developments of the thin plate spline was done by Duchon, [Ref. 24]. Analogous to the polynomial spline, it was developed to fit a surface to certain points by application of convenient localized weight.

The thin plate spline basis is defined as follow:

$$Q(x,y) = \sum_{k \in I} [A_k d_k^2 \log d_k + a + b_x + c_y] \qquad (4.11)$$

where

$I = \{k : Q \text{ is to take on the value } f_x \text{ at } (x_k, Y_k)\}$

$d_k^2 = (x-x_k)^2 + (y-y_k)^2$ and the coefficients A_k, a, b, and c are determined by the solution of the following linear system.

(4.12)

$$\sum_{k \in I} A_k d_k^2 \log d_k + a + b_x + c_y \Big|_{(x,y) = (x_i, Y_i)} = f_i \quad i \in I$$

$$\sum_{k \in I} A_k = 0$$

$$\sum_{k \in I} A_k x_k = 0$$

$$\sum_{k \in I} A_k Y_k = 0$$

This system of equations has some interesting properties: it is symmetric but not positive definite. The three last relations of equation 4.12 has a geometric effect that suppress all terms that grow faster than linearly as the distance of the interpolation points is increased.

The Figure 4.3 shows the rectangular regions covered by the thin plates. Each subrectangle is covered by at least four thin plates and the final value of each point (node of the uniform square grid) is the average of the four local computations.

The implementation of this algorithm can be seen in Appendix A and the Fortran listing is provided in Appendix C.

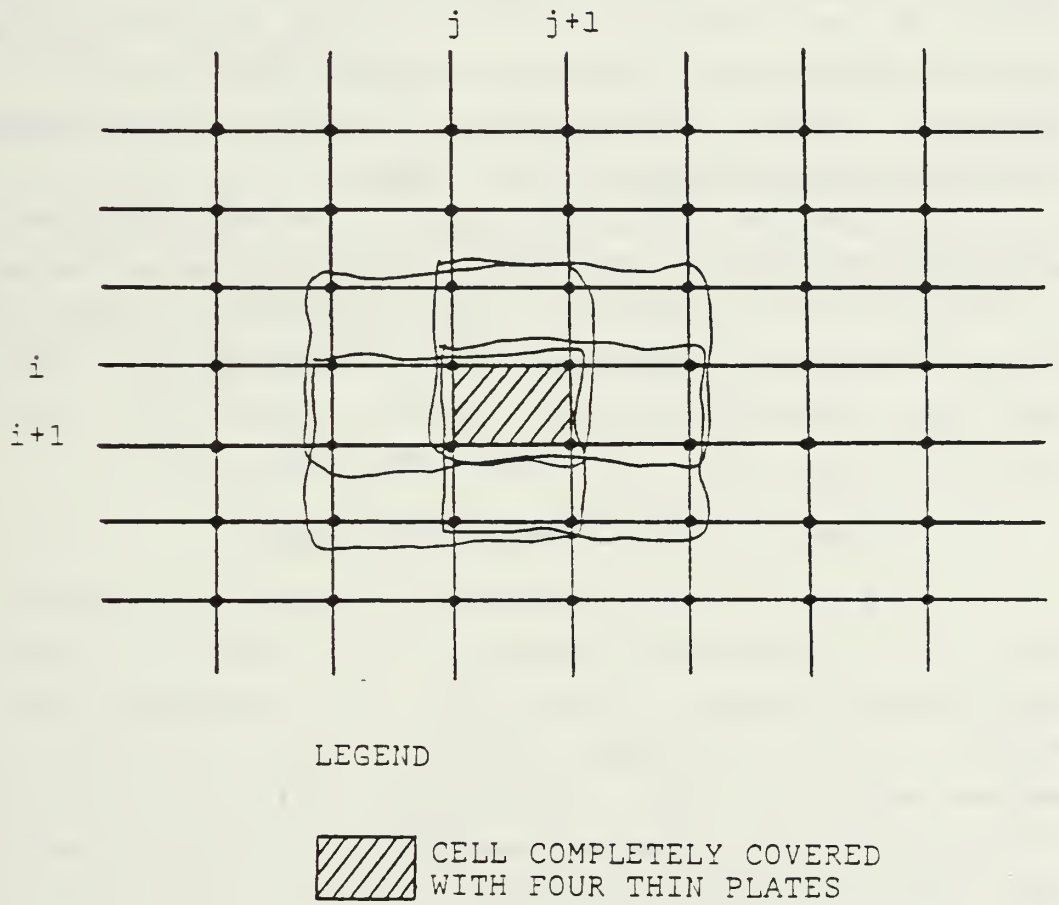

Figure 4.3 Rectangular Regions Covered by the Thin Plates.

D. TRIANGLE-BASED INTERPOLATION METHOD

1. Introduction

The triangle-based interpolation method was chosen as an alternative approach to reconstruct the compressed image generated in Chapter III. The rationale to select this algorithm is that it is conceptually different from the thin plate spline method seen in the previous section, and it can be used for comparison purposes. In Chapter V, the testing phase, the results from both methods will be analyzed and compared for identical set of input data.

The algorithm is conceptually simple. It consists of two main steps. In the first step it builds a triangular covering the convex hull of the given set of (x_i, y_i) data using the (x_i, y_i) data points as vertices of the triangular cells. In the second step it evaluates an arbitrary point (x,y), in our case a (256x256) uniform squared grid, by just doing a linear interpolation inside the correspondent triangle where each point (x,y) lies.

As we can anticipate, this method generates a C^0 surface on the space. The Figure 4.4 shows the projection of a typical set of triangles in the x,y plane, where x and y are the spatial coordinates. The triangulation algorithm used here was implemented by Franke, [Ref. 25], using the ideas of Akima, [Ref. 17], and Lawson [Ref. 18].

2. Outline of the Triangulation Method

A step by step procedure to construct a uniform grid over a set of scattered points(x,y) is given by Lawson in a comprehensive paper, [Ref. 18]. We transcript here the outline of the proposed method: "The criterion of the maximum angle is used. According to this criterion when a set of four points used as vertices of a quadrilateral with each internal angle smaller than π , i.e., the quadrilateral

is convex. One chooses out of two possible ways of partitioning the quadrilateral into a pair of triangles. The partitioning that minimizes the minimum interior angle of the two triangles will be selected.

In triangulating the x,y plane, we first connect the closest pair of points. We next add a point sequentially in ascending order of the distance from the midpoint of the closest pair of points. This ordering in adding new points assures that a new point to be added always lies outside the polygon constructed with the old points. The new point lies outside the circle that is centered at the midpoint of the closest pair of points and passes through the most lately added old point while the polygon lies inside the circle. Each time a new point is added we construct trinagles by connecting the new point with the old points that are visible from the new point and, whenever necessary, "exchange" triangles.

The second step of the process consists of linear interpolation of each point (x,y) of the uniform square grid output with the correspondent triangle constructed in step one. In Appendix A some comments on the computer implementation of this algorithm is presented, and in Appendix C a program listing is provided.

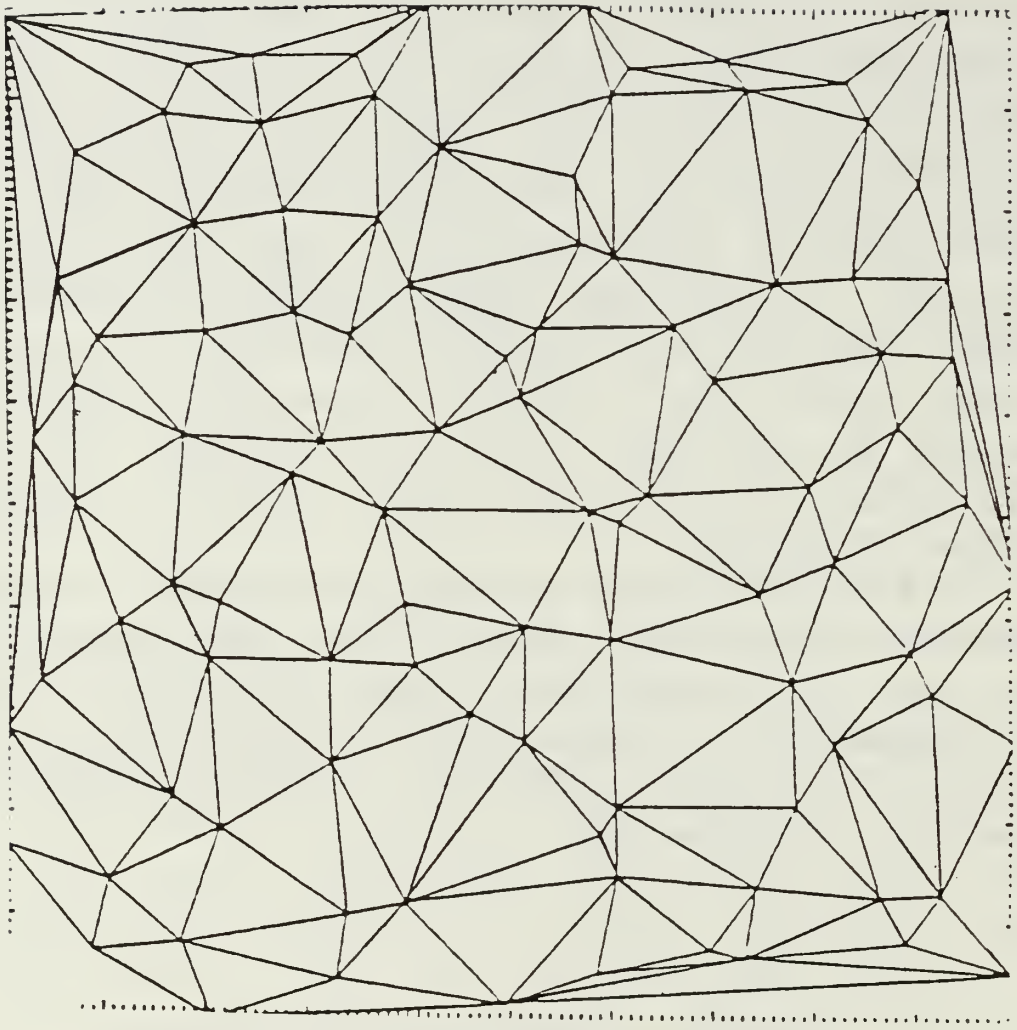

Figure 4.4 Triangulation Grid.

V. TESTS, MEASUREMENTS AND CONCLUSIONS

A. INTRODUCTION

In Chapter I the general concepts of digital image representation and processing were presented. In Chapter II an overview of the spline functions and particularly the B-spline representation were done. The mathematical properties and the relevant characteristics which make the B-spline a convenient and a powerfull tool to work within image processing were carefully explored there. In Chapter III the mathematical framework developed in Chapter II was used to solve our problem of generating data compression by selecting uneven knots from the contours of a given digital image. The highlights of the algorithm used there were that the knots are selected in an automatic fashion, so that a trade-off between smoothness and closeness of fitness is achieved for each contour. The relation among neighboring contours, namely three- dimensional effect, is accounted by using a weighting function that is related with the density of contours in the vicinity of each contour point. The development in Chapter IV was towards the reconstruction of the image from a set of scattered points generated in Chapter III. Two different approaches, namely triangulation and thin plates spline, were used in the reconstruction phase.

In this chapter now, a series of experiments is carried out to validate the whole process developed in the former chapters. A set of four selected scenes covering a broad range of features are tested. A criterion for acceptance of the process based on performance, reliability, and robustness is presented. The sensitivity analysis of the system,

with regard to the various user's controlled parameters, is done and the results are checked against our acceptance criterion.

B. SENSITIVITY ANALYSIS

1. Objective

The rationale for a study of the sensitivity of the whole image processing system with respect to the user's controlled parameters is the following. We can, based on our acceptance criterion to be developed, judge how good is our model. As we have seen in Chapter I, our goal is to achieve data compression within a reasonable processing time and reconstruct the image within acceptable error compared with the original one. The factors that influenced the performance of the image processing system are the intrinsic characteristics of the algorithm described in the former chapters, and the user's controlled parameters which are the subject of our study here.

2. Acceptance Criterion

A desirable image processing system should have the following attributes:

Generality : the system should be applicable to a variety of images or at least to a complete class of images.

Predicability : given an image to be processed, the performance of the system and the characteristic of the output product can be anticipated based on the previous knowledge of the system.

Robustness : the algorithm which the system is based upon should be strongly formed. It should have a solid mathematical framework over which some heuristic approaches can be adopted.

Performance : the processing should be accomplished in a reasonable time with the required precision.

Simplicity : the system should be simple to understand and simple to operate.

Modularity : the technique, divide and conquer, should be used to design and implement this digital image processing system. A system easy to understand, expand, and maintain can be derived using the modular approach.

Gracefull degradation : an image processing system should not collapse with the small anomalies or pseudo anomalies in the image to be processed. It should have alternatives to patch small perturbations on the system automatically. For example, in the process of finding the contours we have two categories: the opened contours and the closed contours. Eventually, if we have a saddle shape region, we can end up with a contour that it is neither opened or closed. The algorithm should be able to handle this problem in such a way that it does not crashes the program and will not have unexpected result. Another case is in the evaluation of the roots of an equation. If the limited number of iterations is reached before the required precison is obtained, a compromise solution should be achieved instead of occurrence of a catastropiic failure. Like these two examples we can list many others. Many of these goals are in conflict with one another, this trade-off is a necessary action to be taken.

Next, the input variables (user's controlled parameters) will be described followed by the output parameter. The sensitivity of the system will be studied based on the variation of the input variables.

3. User's Controlled Parameter

In order to simplify the sensitivity analysis of the image processing system developed in this work, it is

advantageous to explore its modular organization and divide it into stages. The natural division at the highest level is to have two phases. The first phase concernes the data compression and the second deals with the image reconstruction from the compressed data set. The first phase can be further subdivided into the contouring phase and the knot selection phase. With this structure in mind, we can study the user's controlled parameters and their influence in each phase of the process.

a. Contouring Phase

In this phase the contours are extracted from a given image represented by a uniform (256x256) square grid. The parameters and actions under the user's control are:

i) Filtering : a low pass filter is applied on the raw image for smoothing purpose. As we know, we may have an image contaminated with noise, and particularly this is the case of aerialphotographic pictures. The causes of this spurious effect that may appear in digital image are due to poor sampling system or transmission channel. In our case here, a moving average filter is applied by the user who can optionally do the filtering operation or bypass it. The filtering operation is the first possible treatment on the signal (input image).

ii) Contour layers : the number of sections in which the image will be contoured is called "NC". According to the constraints listed in Appendix A, this parameter can vary in the range from two to 20. In practice, as we will see later on in this chapter, a typical value for NC is ten. Small values tend to introduce large error in the reconstructed image and large values tend to degenerate the compression ratio.

iii) Threshold : this parameter is called "THRES". It fixes the elimination threshold that scrappes

the contours with less points than THRES. This parameter like the filtering has a smoothness effect. The difference is that it acts after the contours are generated while the filtering option is applied before. Besides its low pass filtering characteristic, the "THRES" parameter can reduce the amount of data to be stored. Therefore, it is a positive factor if we want to have high compression ratio. On the other hand, the low pass filter has side effects; removal of the high frequencies in a signal means degrading the sharpness of the edges. Thus, we must trade off these features for better results achievement. The range from five to 10 for "THRES" has demonstrated to be good.

b. Knot Selection Phase

This phase corresponds to the second step towards the data compression, and constitutes the main core step of the image processing system here. The algorithm to perform the knot selection task was developed in Chapter III. Its main objective is to extract the "strategic" knots of the contours generated in the first phase. Intuitively, the idea is to choose more knots in regions where the contours bend with small radius of curvatures (planar effect) and/or in regions where the density of the contours is high (three-dimensional effect). Mathematically the problem is a minimization of a function under constraint, i.e., a Lagrange type of problem. In this phase there are six parameters under the user's control which are discussed below.

i) Reference weight ($W0$) : it is assigned to each contour point (x,y). The typical value for $W0$ is one.

ii) Incremental weight ($\triangle W$) : it gives each contour point the weight according to the density of contours crossing the neighborhood of that referred point.

iii) Neighborhood span (SPAN) : it defines the size and shape of the neighborhood where the contour density measurements will be performed. In order to simplify the implementation, a squared region centered at the point under measurement is chosen. The size of the square is two times the SPAN plus one (see Figure 3.4), and the typical value for the SPAN is five.

iv) Smoothing factor (S) : it is the parameter that sets the trade-off between smoothness on closeness of the curve fit. The two bound values for S are "S = 0" that results in the interpolation polynomial with no data compression at all, and "S = ∞ " which generates the least square polynomial with the theoretic maximum of data compression. The factor S can be adjusted continuously between the upper and lower bound. It is chosen using an heuristic approach that gives an acceptable fitness without geopardizing the compression ratio and the computing time. It is known that the factor "S" ought to be an increasing function of the number of points (x,y) in the contour being worked out. Several authors, Dierckx [Ref. 15], Reinsch [Ref. 27], have proposed empirical ways to choose the factor S. Here in this development we are going to use a generating function for S given by equation 3.35. This has demonstrated good fitness results for contours with both small and large number of points, while keeping a quite reasonable computing time and producing a substantial data compression ratio.

v) Tolerance (TOL) : this parameter sets the tolerance by which the root of the equation 3.23 should be evaluated. Once our resolution is the space between two adjacents nodes of a uniform square grid (256x256), there is no advantage in using a small value for TOL because all selected knots will be round off to the nearest node on the grid. Besides, the computing time is an inverse function of TOL. A typical value for TOL is TOL = 0.001.

vi) Maximum number of iteration (MAXIT) : this parameter sets the upper limit for the number of iteration in the solution of equation 3.23. This parameter is also related with the computing time during this second phase. A typical value is MAXIT = 20.

c. Reconstruction Phase

In Chapter IV two different image reconstruction approaches were used, namely triangulation method and thin plate splines method. The first one does not require any input parameter from the user except the files of input data generated in the previous phase, as it is illustrated in Figure A.1, whereas the second one requires an extra parameter, namely "NPPR" to be supplied by the user. This parameter is an estimate of the number of points per region (defined in Chapter IV). The recommended value for this parameter is in the range from eight to 30.

So far we have described ten user's controlled parameters and the respective ranges. It is easy to realize that for a set of different images combined with a set of ten input parameters setting we certainly would end up with huge number of testing cases. Therefore, before discussing the output parameters and the test itself, it is worthwhile to narrow the variability of the variables. Using the past experience and some reasoning, we can reduce the degree of freedom by fixing some of the described parameters. The following parameters were chosen to be kept constant.

parameter	status/value
FILTERING	yes
SPAN	5
S	$\sqrt{M}$ (M MOD 100 + 1)
TOL	0.001
MAXINT	20

where M is the total number of points per contour and the other names were defined above.

Thus, the effective varying parameters which will be varied in the overall procedures are: NC, THRES, WO, ΔW, NPPR.

4. Output Measurements

Once the user's controlled parameters are presented, it is time now to introduce the measurements. The sensitivity of these measurements with regard to the paramenters is to be studied. Those measurements are: CPU time in each phase of the process, storage required, compression ratio achieved, and qualitative and quantitative error measurement between the original and the reconstructed image. With the definitions and assumptions set we are ready to start the testing that is the subject of the next section.

C. TESTS AND RESULTS

Four aerialphotographical images were selected to test and verify the image processing model. These test sets were chosen to cover a wide range of cases, i.e., from a very doll image to a heavy busy one in order to check the acceptance criterion defined in the previous section.

For each of these images the whole procedure, i.e., contouring, knot selection, and reconstruction defined in the earlier section, were recorded and the results appear in Tables I to VIII and in Figures 5.1 to 5.24.

The Figure A.1 shows the diagram of the image process system, where the three phases are represented and so the interaction between programs and files. As we have mentioned in Chapter I, the test images are represented by a uniform square grid with spatial coordinates ranging from zero to 255, and this same range is used for the gray level quantization.

Let's now summarize the notation used in the tables of the test cases.

i) First Phase : Contour Generation

FILE ID : identification of the data file under test.

NC : number of contour levels in which the input data will be divided.

THRES : threshold that defines the contours to be purged.

TOTCONT : total number of contours generated which meet the threshold value.

TOTPAIR : total number of coordinates pair (x,y) representing the total number of contours (TOTCONT).

MINLEVEL: minimum gray level value found in the given data, ranging from zero to 255.

MAXLEVEL: maximum gray level value found in the given data, ranging from zero to 255.

CPUTIME : total of CPU time in minutes required to accomplish the current phase of the image processing.

ii) Second Phase : Knot Selection

EXP : experiment number.

WO : reference weight.

ΔW : incremental weight.

N : number of selected knots, (x,y) pairs.

N' : effective number of selected knots.

C_R : compression error achieved (see Appendix B for details).

TCOM: total time required for the compression phase (contour generation and knot selection).

iii) Third Phase : Image Reconstruction

TRIA : number of triangles generated.

E_{RMS} : root mean square error.

E_{REL} : relative error.

NPPR : estimate of number of points per rectangle.

The four study cases will be presented next. The data for each case comes in the following sequence:

- two tables containing the parameter setting and measurements of each phase.
- One picture of the original image.
- Three contour pictures showing the contour map generated in the first phase, followed by two contour maps generated in the second phase for two different parameters setting.
- Two pictures of the reconstructed image, one for each method, triangulation and thin plates respectively.

TABLE I

Test Case I - First Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
TAD.DAT		10		5		356		19351		62		210		8	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0.2	6014	16	4142	15.82	24	14	18	8.63	5.59	8205	25	8.74	5.96
2	1.0	0.2	6014	16	4142	15.82	24	20	21	7.78	5.04				
3	0.5	0.3	4575	12	3214	20.39	20	14	15	8.30	5.37	6387	14	8.87	5.75
4	0.5	0.3	4575	12	3214	20.39	20	20	17	8.25	5.31				
5	1.0	0.0	4425	11	3223	20.33	19	15	16	8.12	5.27	6406	14	9.04	5.85
6	1.0	0.0	4425	11	3223	20.33	19	20	20	8.09	5.24				
7	1.0	0.1	5244	13	3698	1772	21	15	17	8.06	5.22	7356	20	8.81	5.70
8	1.0	0.1	5244	13	3698	1772	21	20	19	8.00	5.15				
9	1.2	0.2	6990	19	4570	14.34	27	15	19	7.70	5.02	9082	35	8.66	5.60
10	1.2	0.2	6990	19	4570	14.34	27	20	22	7.68	5.00				

TABLE II

Test Case I - Second Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
TAD.DAT		12		10		319		22481		62		210		9	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0.0	4991	13	3773	17.37	22	17	18	7.66	4.96	7484	20	8.54	5.52
2	1.0	0.0	4991	13	3773	17.37	22	23	23	7.65	4.95				
3	1.0	1.0	12363	50	6293	10.41	59	17	26	7.62	4.93	12527	62	7.82	5.06
4	1.0	1.0	12363	50	6293	10.41	59	20	28	7.82	5.06				
5	0.5	0.2	4750	13	3502	18.71	22	14	16	7.93	5.12	6982	17	8.45	5.47
6	0.5	0.2	47.50	13	3502	18.71	22	17	17	7.83	5.08				
7	1.2	0.2	8334	37	5487	11.94	48	14	33	7.48	4.93	10909	44	7.67	4.97
8	1.2	0.2	8334	37	5487	11.94	48	20	37	7.44	4.81				
9	0.8	0.2	6153	26	4357	15.04	35	14	29	7.90	5.10	8677	29	8.07	5.32
10	0.8	0.2	6153	26	4357	15.04	35	20	31	7.88	5.07				

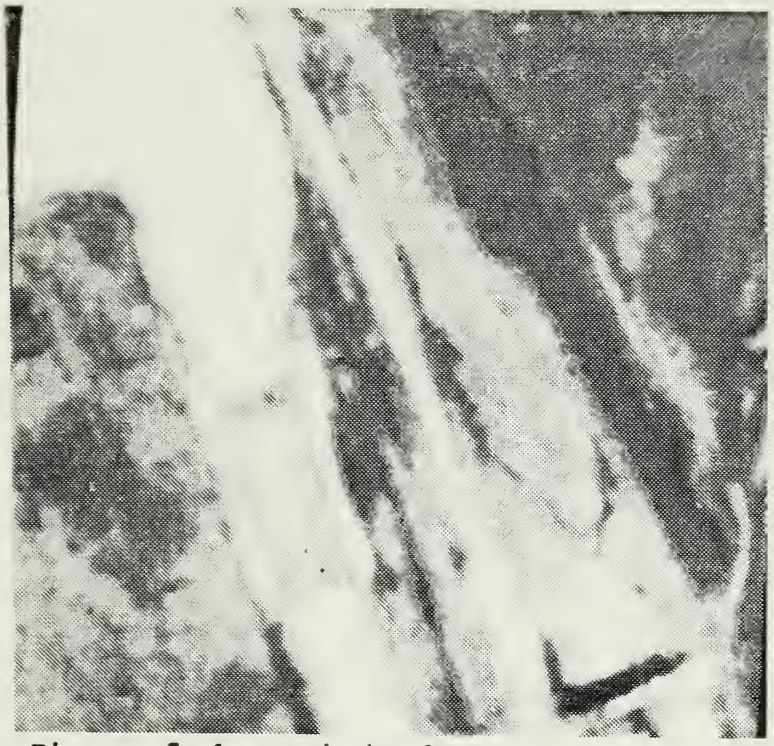

Figure 5.1 Original Image - Case I.

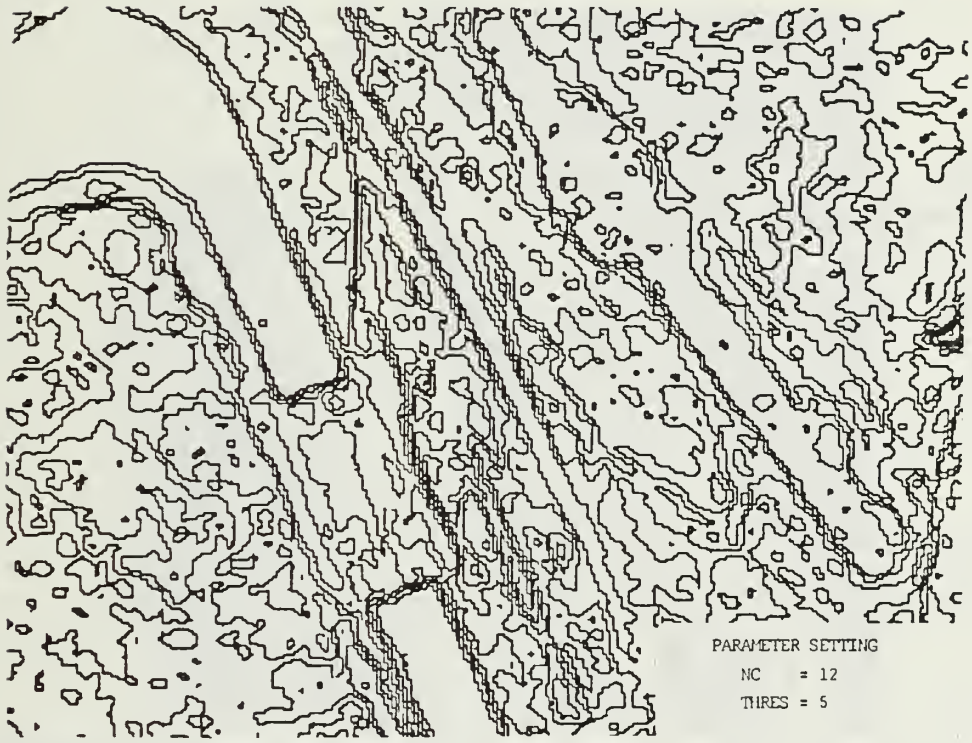

Figure 5.2 Contour Repres. Before Knot Selection - Case I.

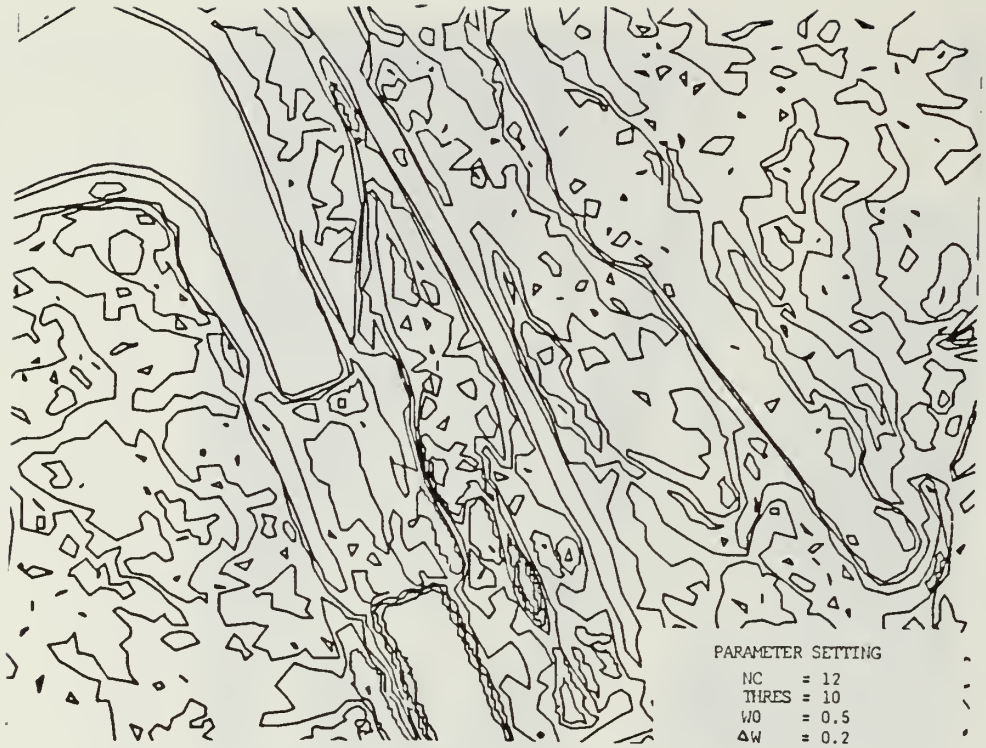

Figure 5.3 Contour Repres. After Knot Selection - Case I.

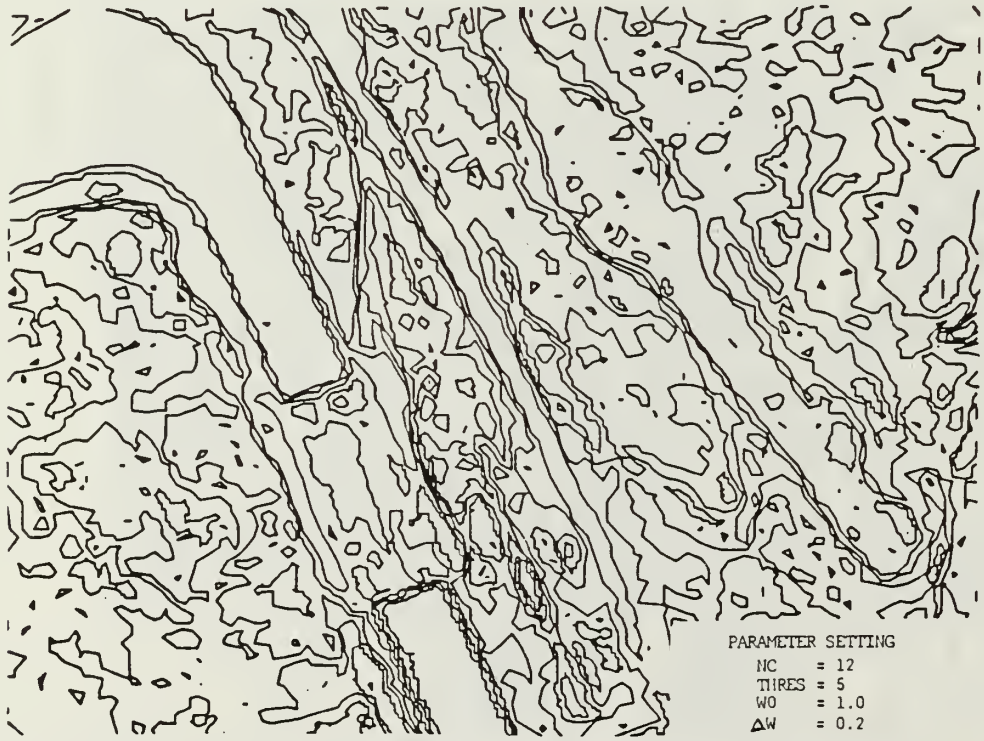

Figure 5.4 Contour Repres. After Knot Selection - Case I.

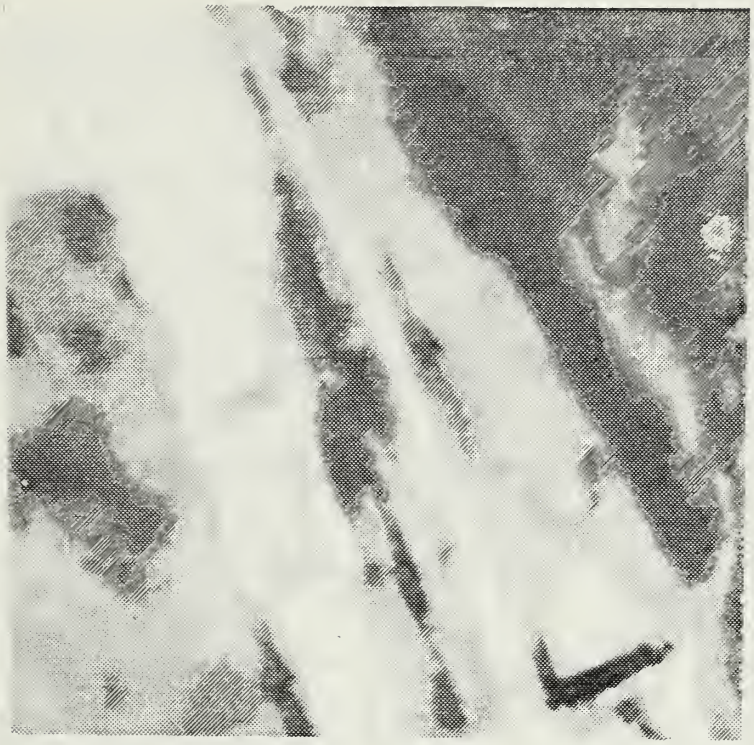

Figure 5.5 Reconstructed Image Via Triangulation - Case I.

Figure 5.6 Reconst. Image Via Thin Plates Spline - Case I.

TABLE III

Test Case II - First Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
CES.DAT		10		5		179		12011		73		210		7	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0	2525	11	1803	36.35	18	12	19	4.62	3.23	3531	12	4.92	3.44
2	1.0	0	2525	11	1803	36.35	18	17	24	4.48	3.14				
3	1.0	0.2	2827	13	2015	32.52	20	14	20	4.51	3.15	3955	13	4.82	3.37
4	1.0	0.2	2827	13	2015	32.52	20	20	27	4.41	3.08				
5	1.0	1.0	3805	17	2422	27.06	24	14	28	4.51	3.15	4771	16	4.68	3.27
6	1.0	1.0	3805	17	2422	27.06	24	20	28	4.51	3.16				
7	1.2	0.4	3611	16	2429	26.98	23	14	23	4.34	3.04	4797	15	4.71	3.47
8	1.2	0.4	3611	16	2429	26.98	23	20	28	4.30	3.01				
9	0.8	0.2	2363	9	1778	36.86	16	14	20	4.47	3.13	3350	9	4.96	3.47
10	0.8	0.2	2363	9	1778	36.86	16	20	27	4.46	3.12				

TABLE IV

Test Case II - Second Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
CES.DAT		15		5		288		18721		73		210		17	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0	4052	17	2873	22.81	34	14	24	3.31	2.31	5693	19	3.85	2.69
2	1.0	0	4052	17	2873	22.81	34	20	29	3.28	2.30				
3	1.0	1.2	8303	43	4543	14.43	60	14	30	3.27	2.28	9039	51	3.86	2.70
4	1.0	1.2	8303	43	4543	14.43	60	20	34	3.23	2.26				
5	1.0	0.8	7379	33	4336	15.11	50	14	31	3.26	2.27	8624	48	3.97	2.78
6	1.0	0.8	7379	33	4336	15.11	50	20	35	3.20	2.24				
7	1.2	0.6	7517	35	4469	14.66	52	14	28	3.27	2.28	8874	44	3.58	2.50
8	1.2	0.6	7517	35	4469	14.66	52	20	30	3.29	2.29				
9	0.8	0.2	4305	20	3005	21.81	47	14	29	3.37	2.35	5942	22	3.59	2.51
10	0.8	0.2	4305	20	3005	21.81	47	20	30	3.39	2.34				

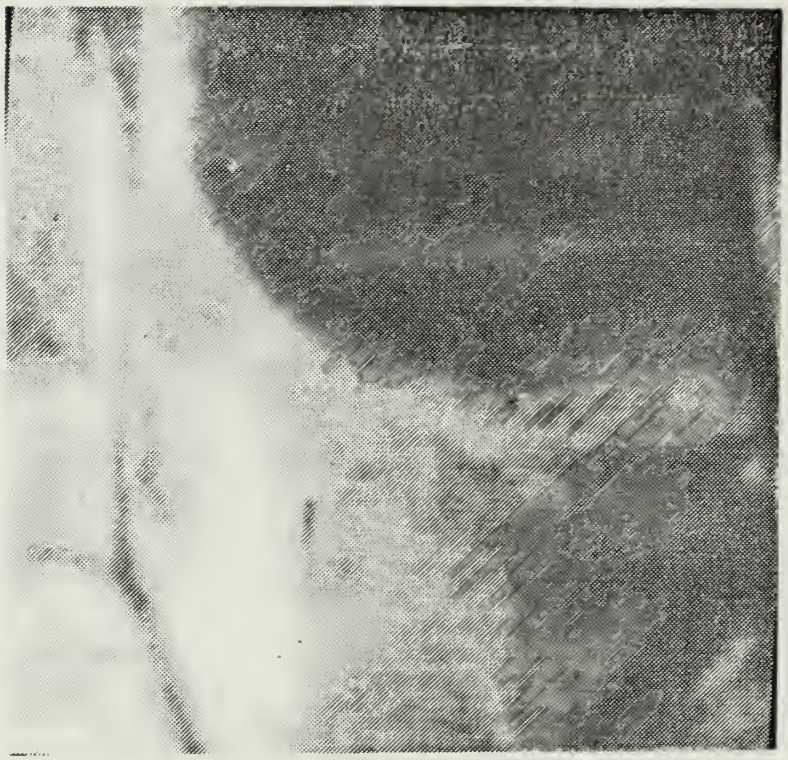

Figure 5.7 Original Image - Case II.

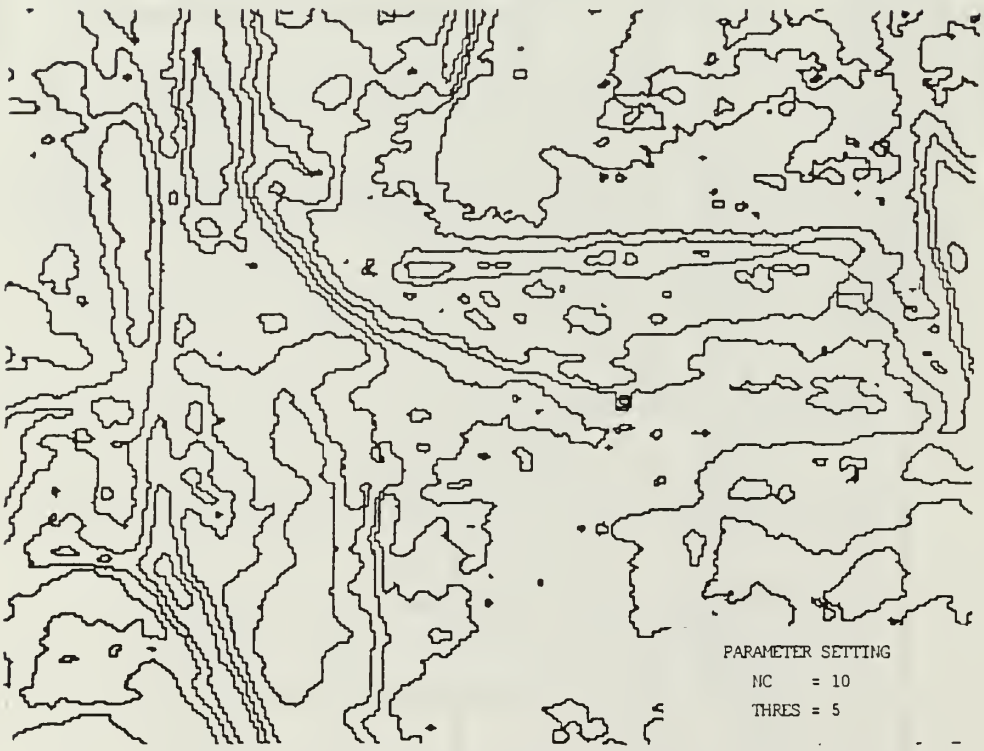

Figure 5.8 Contour Repres. Before Knot Selection - Case II.

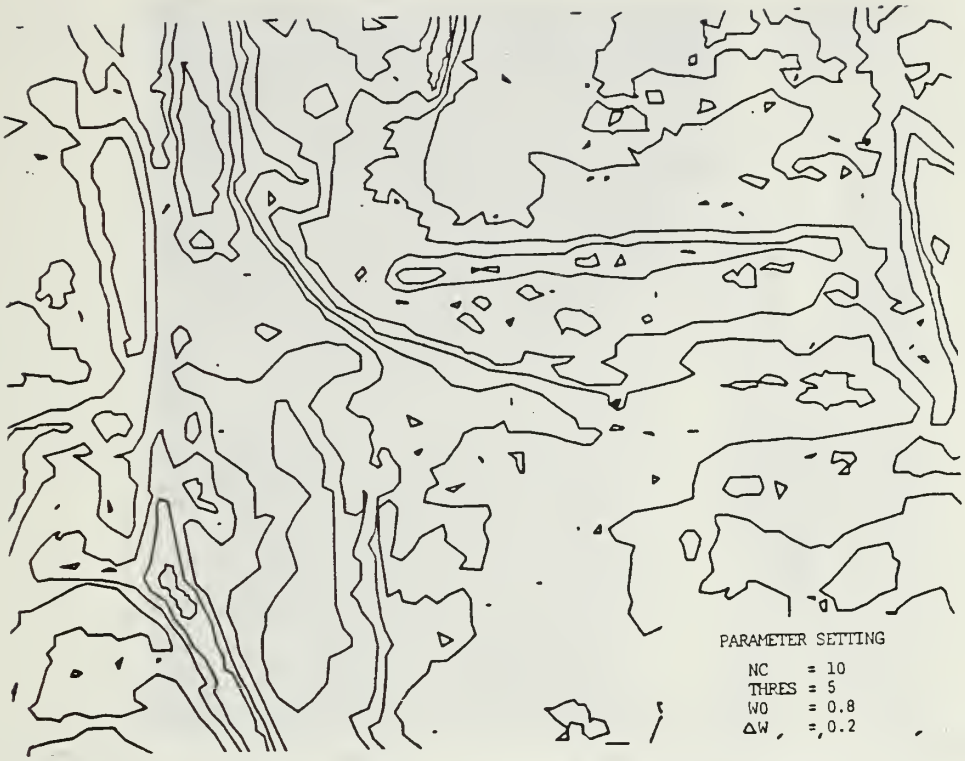

Figure 5.9 Contour Repres. After Knot Selection - Case II.

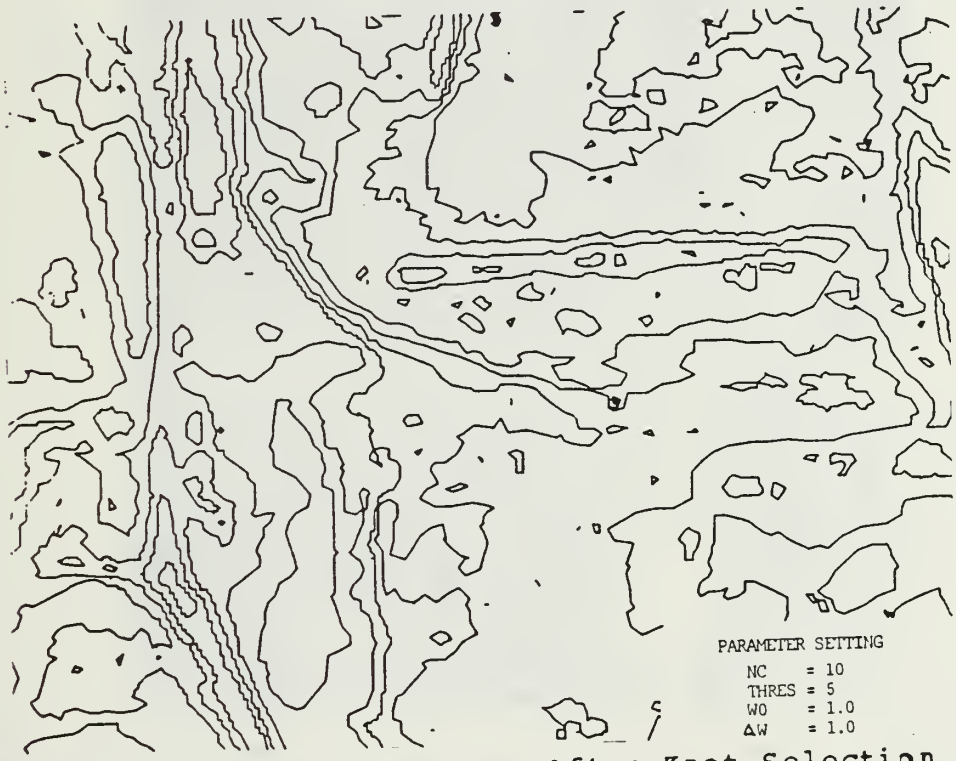

Figure 5.10 Contour Repres. After Knot Selection - Case II.

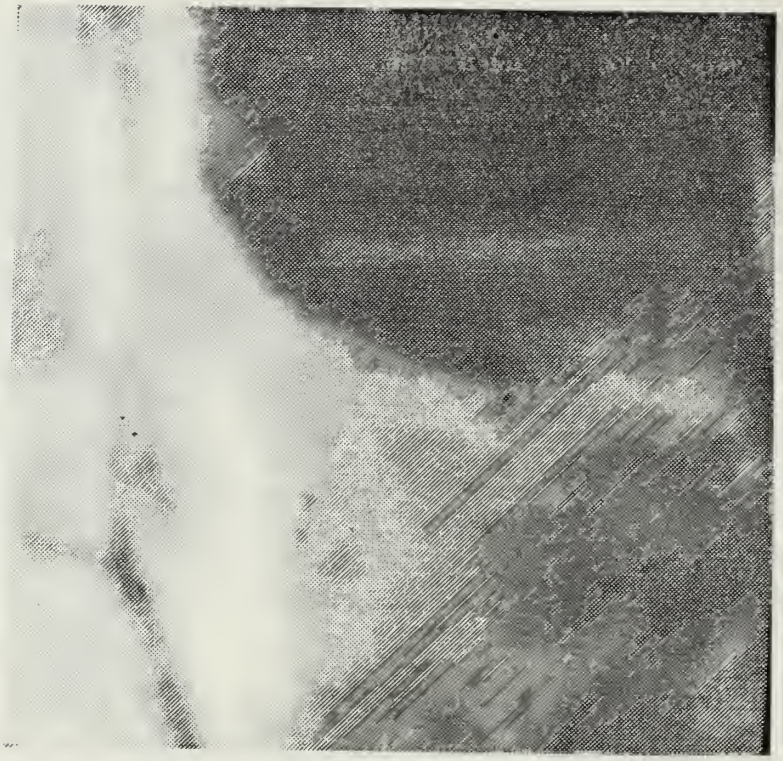

Figure 5.11 Reconst. Image Via Triangulation - Case II.

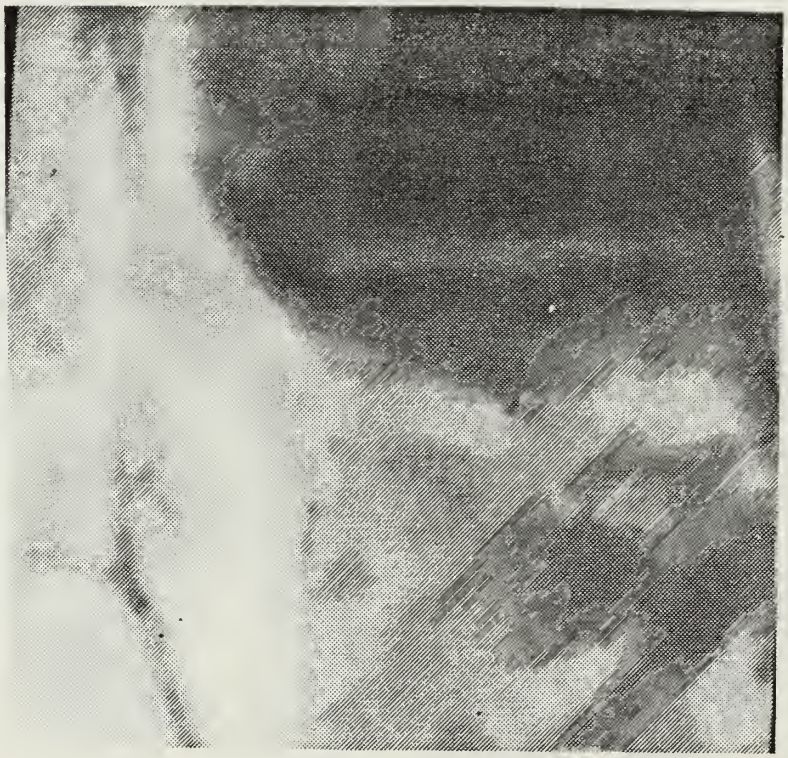

Figure 5.12 Reconst. Image Via Thin Plates Spline - Case II.

TABLE V

Test Case III - First Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
MIR.DAT		10		5		493		23567		52		212		14	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0	5533	21	3921	16.71	35	15	43	12.05	8.23	7765	35	12.06	8.24
2	1.0	0	5533	21	3921	16.71	35	20	45	12.02	8.19				
3	1.0	0.2	8102	29	5084	12.89	43	15	52	11.98	8.16	10084	39	11.83	8.09
4	1.0	0.2	8102	29	5084	12.89	43	20	54	11.94	8.14				
5	1.0	0.4	9754	42	5597	11.71	56	15	59	11.92	8.11	11116	48	11.81	8.07
6	1.0	0.4	9754	42	5597	11.71	56	20	63	11.83	8.09				
7	1.2	0.2	9038	35	5839	11.22	49	15	67	11.73	8.05	10910	45	11.66	7.97
8	1.2	0.2	9038	35	5839	11.22	49	20	69	11.70	8.01				
9	0.8	0.1	5714	21	4015	16.32	33	15	46	12.04	8.24	7919	26	12.03	8.23
10	0.8	0.1	5714	21	4015	16.32	33	20	47	12.03	8.21				

TABLE VI

Test Case III - Second Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
MIR.DAT		15		5		807		36089		52		210		18	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0.0	8615	30	5961	10.99	48	15	45	11.09	7.53	11804	91	11.06	7.56
2	1.0	0.0	8615	30	5961	10.99	48	20	47	11.03	7.47				
3	1.2	0.2	15152	37	9472	6.92	55	15	67	10.64	7.09	17315	122	10.35	6.93
4	1.2	0.2	15152	37	9472	6.92	55	20	69	10.59	7.03				
5	1.0	0.2	13710	32	8915	7.35	50	15	62	10.83	7.42	16250	101	10.52	7.20
6	1.0	0.2	13710	32	8915	7.35	50	20	66	10.78	7.37				
7	1.0	0.1	11214	31	7251	9.04	49	15	51	10.95	7.49	14386	80	10.56	7.25
8	1.0	0.1	11214	31	7251	9.04	49	20	55	10.91	7.45				
9	0.8	0.2	12069	30	7526	8.71	48	15	53	11.04	7.52	14961	85	11.21	7.66
10	0.8	0.2	12069	30	7526	8.71	48	20	56	10.99	7.49				

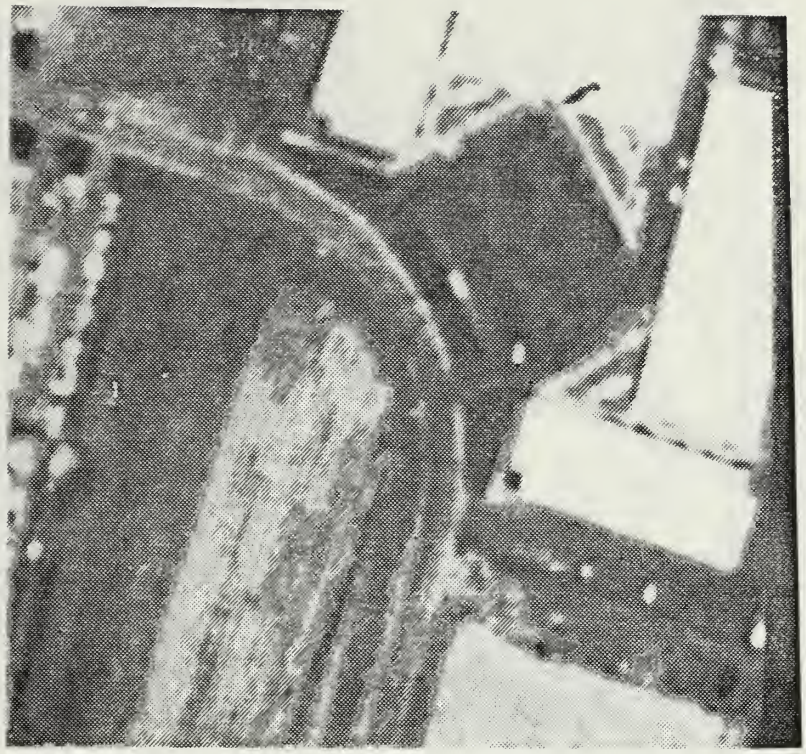

Figure 5.13 Original Image - Case III.

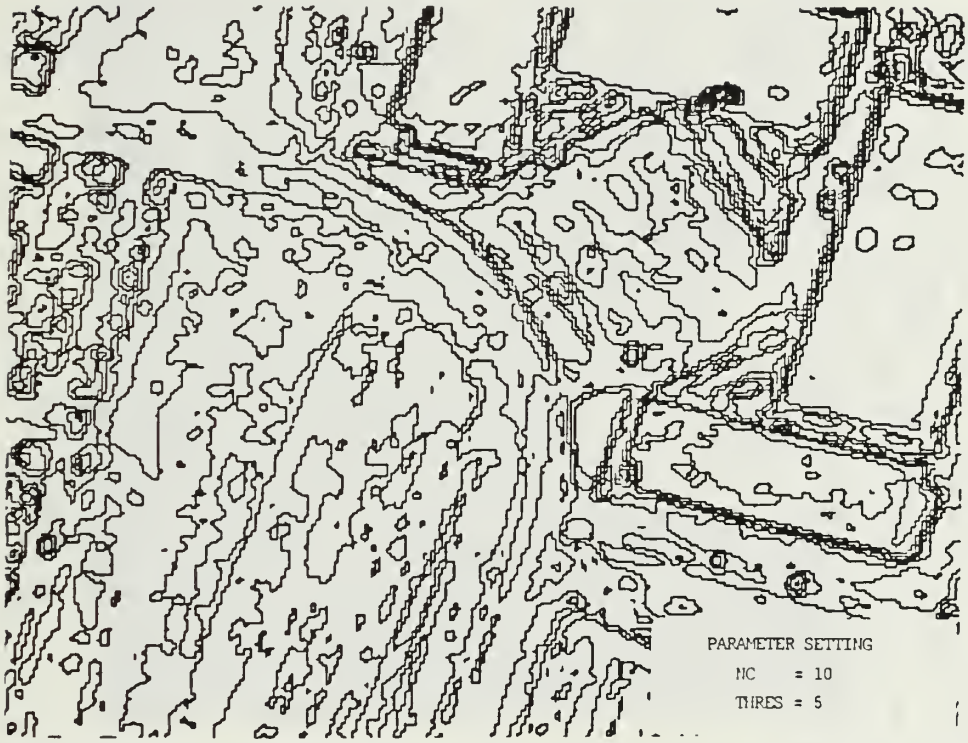

Figure 5.14 Contour Repres. Before Knot Selection - Case III.

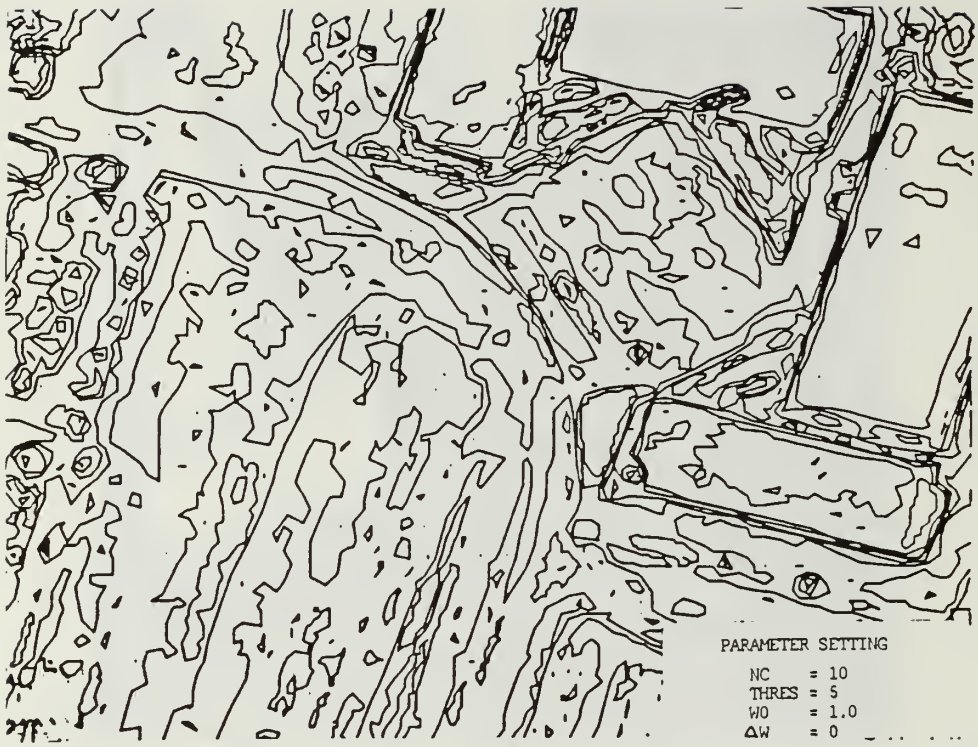

Figure 5.15 Contour Repres. After Knot Selection - Case III.

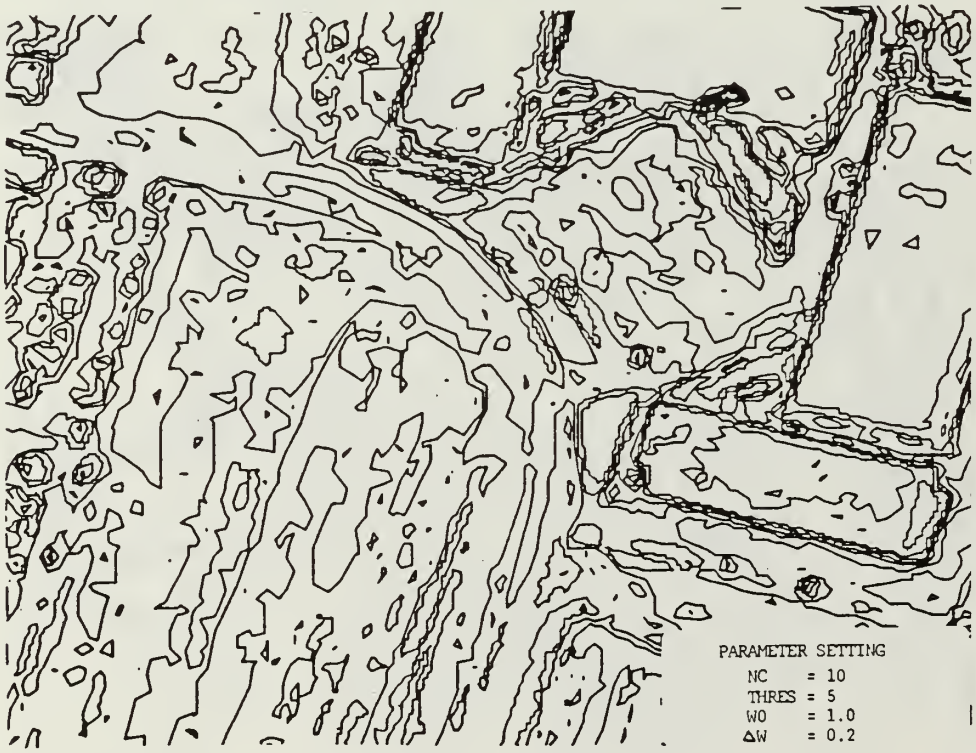

Figure 5.16 Contour Repres. After Knot Selection - Case III.

Figure 5.17 Image Reconst. Via Triangulation - Case III.

Figure 5.18 Reconst. Image Via Thin Plates Spline - Case III.

TABLE VII

Test Case IV - First Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID	NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME		
APA.DAT	10		5		1055		36793		8		224		13		
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0	9406	33	6729	9.74	43	14	38	16.21	12.12	13354	103	16.91	12.64
2	1.0	0	9406	33	6729	9.74	43	20	41	16.12	12.07				
3	1.0	0.2	14822	51	9017	7.27	64	14	57	15.92	11.90	17928	180	16.22	12.13
4	1.0	0.2	14822	51	9017	7.27	64	20	59	15.86	11.83				
5	1.0	0.1	12349	42	8115	8.08	55	14	50	15.78	11.80	16125	148	16.32	12.26
6	1.0	0.1	12349	42	8115	8.08	55	20	53	15.39	11.50				
7	0.8	0.4	17398	66	9748	6.72	79	16	63	15.60	11.66	19394	216	16.70	2.16
8	0.8	0.4	17398	66	9748	6.72	79	20	69	15.56	11.63				
9	0.8	0.2	13193	47	8446	7.76	60	16	55	15.84	11.84	16786	158	16.41	12.27
10	0.8	0.2	13193	47	8446	7.76	60	20	57	15.47	11.56				

TABLE VIII

Test Case IV - Second Part

FIRST PHASE: CONTOUR GENERATION															
FILE ID		NC		THRES		TOTCONT		TOTPAIR		MINLEVEL		MAXLEVEL		CPUTIME	
APA.DAT		14		5		1055		52694		8		224		21	
SECOND PHASE: KNOT SELECTION								THIRD PHASE: IMAGE RECONSTRUCTION							
								THIN PLATES SPLINE				TRIANGULATION			
EXP	WO	O	N	CPU TIME	N'	C_R	TCOMP	NPPR	CPU TIME	E_{RMS}	E_{REL}	TRIA	CPU TIME	E_{RMS}	E_{REL}
1	1.0	0	13396	46	9462	6.93	67	16	63	14.90	11.14	18843	202	15.43	11.54
2	1.0	0	13396	46	9462	6.93	67	20	65	14.82	11.08				
3	1.0	0.1	18251	63	11540	5.68	84	16	85	15.59	11.65	23001	354	15.23	11.39
4	1.0	0.1	18251	63	11540	5.68	84	20	87	15.63	11.69				
5	1.0	0.2	22411	72	12974	5.05	93	16	91	14.66	10.96	25869	364	15.24	11.39
6	1.0	0.2	22411	72	12974	5.05	93	20	95	14.62	10.93				
7	0.8	0.3	23950	86	13352	4.91	107	16	93	14.67	10.97	26688	390	15.21	11.37
8	0.8	0.3	23950	86	13352	4.91	107	20	98	14.60	10.92				
9	0.8	0.1	15775	50	10511	6.23	71	16	63	14.93	11.16	21007	258	15.41	11.52
10	0.8	0.1	15775	50	10511	6.23	71	20	66	14.86	11.11				

Figure 5.19 Original Image - Case IV.

Figure 5.20 Contour Repres. Before Knot Selection - Case IV.

Figure 5.21 Contour Repres. After Knot Selection - Case IV.

Figure 5.22 Contour Repres. After Knot Selection - Case IV.

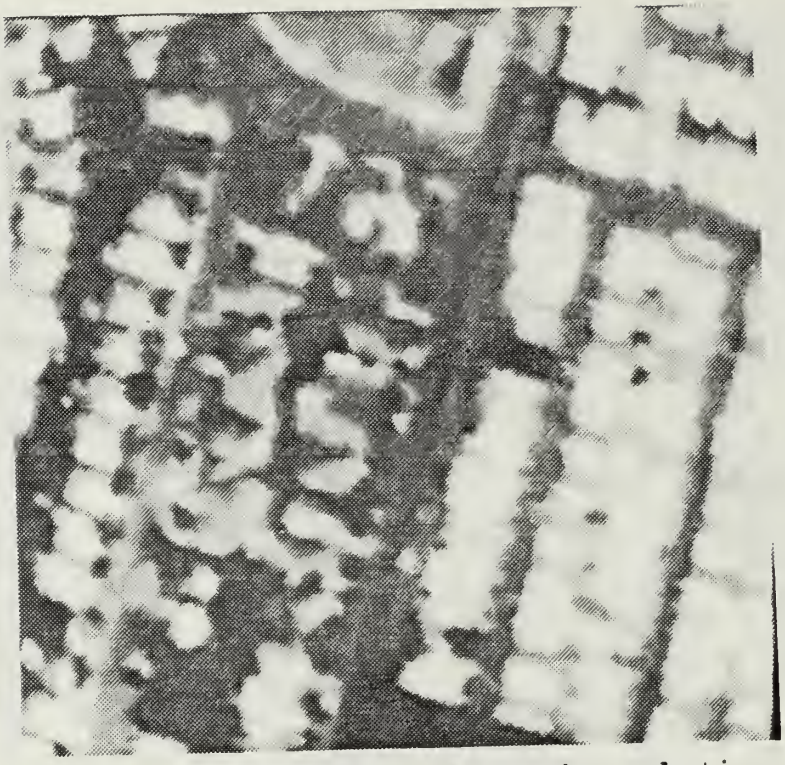

Figure 5.23 Reconst. Image Via Triangulation - Case IV.

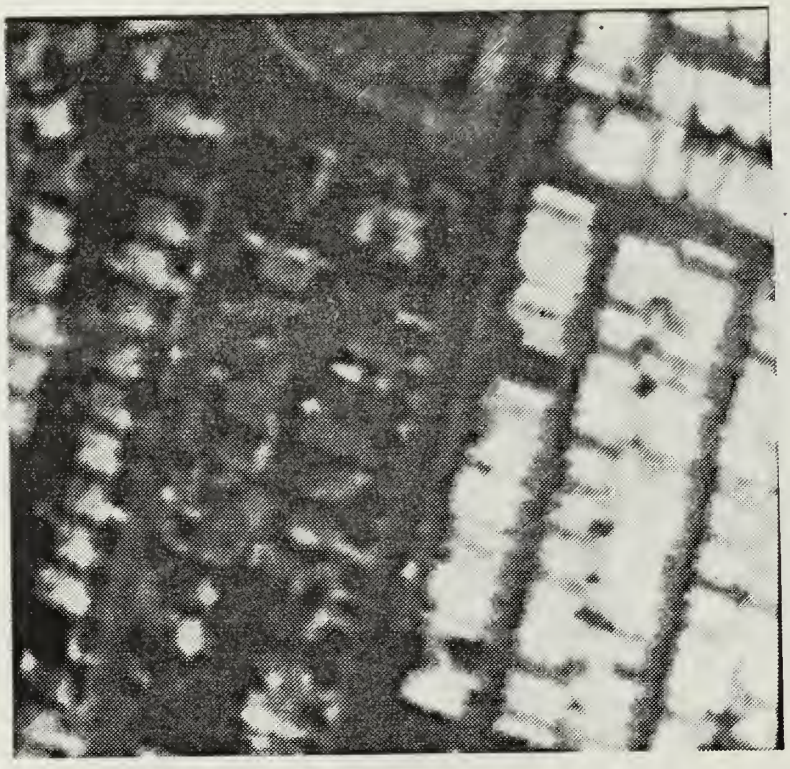

Figure 5.24 Reconst. Image Via Thin Plates Spline - Case IV.

D. RESULT ANALYSIS AND CONCLUSIONS

An hierarchical pattern will be used to analyze the results presented in the previous tables and pictures. First of all the problem will be tackled as a whole. The analysis of the intermediate results, and the performance of each one of the three phases will be done separetely.

The data compression ratio generally depends on the contents of the image. The compression ratio decreases when the image becomes busier. In another words, we have a smaller compression ratio for an image with a large number of features.

In the four study cases presented here the above conclusion can be checked easily. The compression ratio ranged from a value of about 30 in a doll image as in study case II, to a value of about seven in a very busy image as in study case IV.

Compared with the current benchmarks developed by Gonzales, [Ref. 1], and Andrews, [Ref. 4], these compression ratio values are remarkable. In those references, the typical compression ratio achieved was from two to seven and from two to ten respectively.

It is necessary to recall that the coding algorithm used by the two listed authors is different from what we are using here.

With regard to the fidelity of the reconstructed image, as it was expected too, the discrepancy between the qualitative measurement and the quantitative ones (root mean square and relative error) is greater for the busiest images. This can be seen if we compare the case study II, the doll one, against the case study IV, the busy one. For the same setting of parameters we see that the case study IV has a quantitative error of about three times the case study II, while the qualitative measurements in both images give about the same quality for them.

Compared with the results of the above two references, the average quantitative error in the process described here was higher than those ones. Andrews result shows compression error from 0.2% to 6.3% based on their test images. Our result shows compression error from 2.0% to 11.0% on the images used. These errors are based on two different set of test images. It was not found a specific reason for being so. Although if we compare the amount of deterioration of the reconstructed images in a qualitative fashion, there is not too much difference between our samples here and those described in the references.

If we consider the product of compression ratio and quantitative error, and use it as a figure of merit of the image processing system, we will find that the system described here and those of the two references mentioned above have about the same value.

The third factor to be considered is the computing time. In order to get the total time for the image processing system described here, we need to add the three time quantity appearing in the study case tables. The two figures refer to the compression phase itself, and the last one refers to the reconstruction phase. From the result it is clear that the computing time is high, even though an image with (256x256) is considered a large set of data. The processing speed of our computing environment is not the state-of-the art. It should be convenient to remember that the time shown in the table rated as CPU time does include the swapping time required to input and output pages into the physical memory, and it is well known that large array processing generates a huge number of page faults.

As we can see from the tables, the reconstruction method used (phase three) contributes heavily in the total processing time. We will see later on, that depending upon the reconstructed method chosen, the processing time will be

quite sensitive or not to the amount of data to be processed (the selected knots, as it was named before).

Now, that the overall behavior of the system was presented, it is necessary to discuss the performance details of each phase.

The first phase, the contour generation scheme, have demonstrated to be a very efficient and general algorithm. From previous tests, not shown in the presented tables, it was noticed that variations of the threshold (THRES) parameter has minor effect on the output results when THRES lays in the range from five to 10. As a matter of fact, the value of THRES equals to five gives a slight improvement in the qualitative and quantitative error measurements between the output image and the original one. The compression ratio and the computing time have a slight deterioration as it was anticipated. In order to reduce the degree of freedom of the problem, therefore the testing load, we felt that the parameter THRES could be kept constant without undermining the test purposes. Thus, the value of THRES was kept equals to five for all experiments but one recorded in the presented tables.

The number of contours "NC" was the effective varying parameter in the phase one. A range from 10 to 15 was covered for all study cases, and a continuous improvement was achieved in the quality of the reconstructed image for higher values of "NC" with expense of a continuous deterioration of the compression ratio and the computing time. The mentioned range for "NC" has demonstrated to be ideal.

The second phase, the knot selection algorithm, is the core of the image processing system developed here. It gives the strategy to select the knots so that a high compression ratio can be achieved with small loss in the information in the result.

Comparing the contours plots generated in this second phase with those ones produced in the first phase, it is easy to get the feeling on how the algorithm works. Each corner of the contour lines represents a selected knot (output of the second phase), which is clustered in regions where the curve bends with a small radius and/or regions with high density of contour (fast change in gray level).

So far we have discussed the dynamic of the second phase through the analysis of the contours generated there. A natural question to ask is, how does this scheme affect the final product, i.e., the reconstructed image? Clearly, it will inherit the coupled effect of the three phases of the process. In order to isolate the effect of the second phase in the reconstructed image, it is necessary to fix the parameter in both, first and third phases and analyse the final results. This procedure was done many times and the results were recorded in the study case tables presented in the previous section.

From this we can extract the effects of the reference weight "W0" and the incremental weight "ΔW" on the quality of the output image. These two parameters allow a fine tunning of the fidelity of the reconstructed image. A set of value which gives a good trade-off between the error measurements in the reconstructed image and the compression ratio is $W0 = 1.0$ and $\Delta W = 0.1$, which proved to be satisfactory for all four study cases.

In the third phase the two independent approaches, namely triangulation method and thin plates spline method, have yielded interesting results.

The triangulation has demonstrated to be a very general, predictable, robust, and simple method. It covered the four different study cases without any constraint. Regarding to the performance, it has demonstrated to be quite sensitive to the size of the input set (select number of knots). For a

given input data set of about 3500 points, the CPU time is similar to that one required by the second reconstruction method, the thin plates spline method. But as the input number of data points increases, the computing time increases very fast. This is an intrinsic characteristic of the triangulation approach and nothing can be done about it, once a hull of triangles should be constructed over the given points, and the number of triangles is of course an increasing function of the number of points. The act of constructing triangles and working with it becomes very expensive in terms of time if we have a lot of them to deal with.

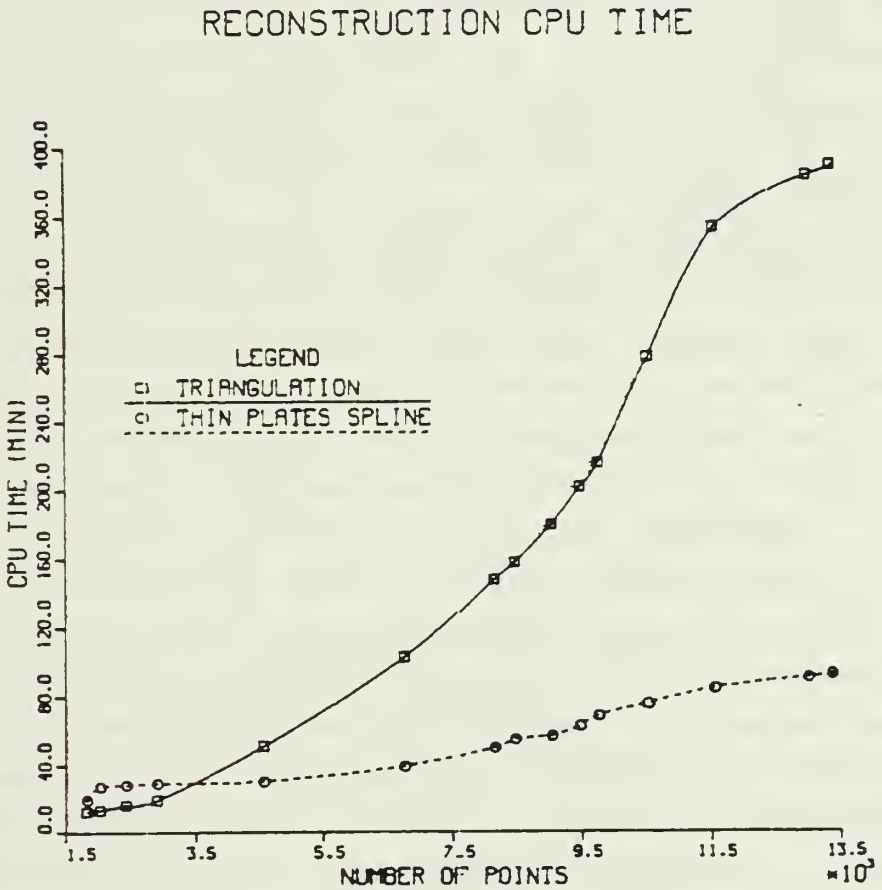

Figure 5.25 Efficiency of the Reconstruction Methods.

In Figure 5.25 is shown the time spent in both reconstruction methods, triangulation and thin plates spline, versus the number of input points to be processed.

For the quantitative error, the triangulation method always showed a higher value than the thin plates spline method, but the difference is not so significative; but for the qualitative measurements, a sort of geometric effects due to the triangulation were present in the images reconstructed through this method.

The second method for image reconstruction, the thin plates spline, did not demonstrate to be quite general as the triangulation method, neither quite predictable in the sense that it can overshoot, and the opposite happens in the triangulation method; and depending upon the special condition on the distribution of the data (region with lack of data), it can generate high overshoot, thus saturating the gray level scale.

Its performance is very good compared with the triangulation method; see Figure 5.25. As we can conclude from the mentioned figure, the thin plates spline method is almost insensitive to the number of input data points, which is a very nice characteristic. The quantitative error, as we said above, is always smaller than that given by triangulation. Conversely to the triangulation method, the thin plates spline method does not have the geometric effect and its fidelity is much better than the first method.

Conversely to the congenital defficiency of the triangulation method, for the thin plates spline method one can get rid of its own drawback, strong overshoot in area with poor distribution of points, by doing some small changes in the program. The specific changes are out of the scope of this work. For more information see Franke's work, [Ref. 19].

For what we have seen in this section, the thin plates spline method is a better option to reconstruct image than

the triangulation method in many aspects, and it is potentially better in an overall sense.

E. RECOMMENDATIONS FOR FUTURE STUDY

The present study was done concerning the future use of some of its contents for real time application in the data compression area. The results achieved in the first two phases, mainly in the knot selection one, were very encouraging. The scheme proved to be very robust and in general it has been demonstrated to degrade gracefully.

As we have seen the process was implemented in a sequential fashion. A parallel processing, although more complex to implement, would decrease the processing time considerably.

Regarding to the third phase, the image reconstruction, only two methods among many possible were tried. It would be worthwhile to study this subject more deeply. A method developed by Lawson, [Ref. 18], would be a good candidate choice.

APPENDIX A
COMPUTING ENVIRONMENT

The algorithms discussed in this section are based on the mathematical developments presented on previous chapters of this text. They consist of a set of computer programs and files which are used as communication interface among programs.

In the next two sections the structure, constraints, and usage of the programs and files are presented in details. These programs were written and/or adapted to the program support environment of the VAX 11/780 and VAX 11/750 machines under the operating systems VMS 3.4. The languages used were Pascal version 2.1 and Fortran version 3.3.

The Figure A.1 shows the interaction between programs and files in a chronological fashion, i.e., the flow indicated by the arrows represents the evolution of the process. The basic elements of such a block diagram are programs represented by rectangles and files represented by hexagons. The ovals mark the beginning and end of the process. Inside each block it was assigned an arbitrary name for programs and files respectively by which they will be called from now on.

The process as it was mentioned before consists of two main distinct phases. In the phase one our concern is to do data compression; from a given set of uniform sampled and quantized data representing an image frame, the contours are extracted and from these contours an algorithm based on the B-spline properties and Lagrange's minimization criteria is applied so that the number of knots necessary to represent the image is drastically reduced, therefore achieving data compression. In the second phase the inverse process is

carried out, i.e., the reconstruction of the image based on the set of compressed data is attempted; this was done through two different approaches namely the thin plates spline method and the triangulation method. The final result is compared with the original one using qualitative and quantitative measurements.

In the next two sections the programs and files which accomplish these tasks are presented.

A.1 PROGRAMS

1. Program Name: WDAT1.PAS

1.1 Objective

Given a packed file MYFILE.DAT containing the image data, this Pascal program unpacks the file and rewrites it in a suitable form to be read by a Fortran program.

1.2 Input Data

File MYFILE.DAT: packed array containing the image frame four times expanded, i.e., a (512x512) grid of pixels.

1.3 Output Data

File IMAG1.DAT: unpacked array containing the image frame converted to an array of (256x256) pixels suitable to be read by a Fortran program.

1.4 Program Structure

This is a Pascal program composed of a single block with no procedures or function calls.

1.5 Constraints and Assumptions

The input file is assumed to be a packed array of (512x512) bytes ranging from zero to 255 with the header stripped off.

1.6 Common Areas

Not applicable in this case.

1.7 Error Messages

No data in the file: if the input file is an empty file.

EOF (end of file): if the file is smaller than expected.

1.8 Usage

```
Compilation: $PAS  WDAT1
Linking    : $LINK WDAT1
Running    : $RUN  WDAT1
```

2. Program Name: THE8M.FOR

2.1 Objective

The image, IMAG1.DAT, consists of an array of (256x256) pixels ranging from zero to 255. This Fortran program extracts the contours from this set of data accomplishing in this way the first step in the data compression phase.

2.2 Input Data

File IMAG1.DAT: unpacked array of size (256x256) containing the pixels of the given image.

File FOR010.DAT: set of parameters which controls the contouring operation. The parameter "NC" is the number of

layers which the image is to be chopped and "THRES" is the threshold that prunes the contours with the number of points below THRES.

2.3 Output Data

File FOR018.DAT: character matrix map. It is an array of character with size (256x256) which represents the projection of the contours with connectivity four on the plane of the spatial coordinates (x,y). The background is represented by the blank character and each layer is represented by an alpha character starting with "A" for the lower level.

File FOR019.DAT: number of contours per each layer and the respective value (gray level) of that layer.

File FOR020.DAT: contour coordinate pairs (x,y).

2.4 Program Structure

This is a Fortran program consisting of 11 subprograms interconnected according to the abstract[1] hierarchical scheme in Figure A.2. The subprograms CONTOUR, SCAN, TRACE, CALC are modified versions of the nonIMSL routine called CONTUR, while the other ones were developed to the specific task described in this text.

2.5 Constraints and Assumptions

The input file IMAG1.DAT is assumed to be an array of integer with dimension (256x256) with gray levels, i.e., pixel brightness value varying from zero to 255.

---

[1]The reason to use the expression "abstract hierarchical scheme" is just to enforce that although Fortran Language has an intrinsic disjoint and nonnesting environment, i.e., a flat structure, we can abstract an hierarchical structure for better understanding.

The maximum number of layers is NC = -20, where the minus sign is used to indicate that the layers are to be chosen automatically by dividing the interval between the maximum and minimum pixel value into the absolute value of NC, rather than with user's defined layer values.

The maximum number of points allowed in each countour is 4800. This figure has been demonstrated more than sufficient during our experiment with aerialphotograph picture; nevertheless, if any contour exceeds this limit the program will stop the contour in that point and an error message will be issued. The program will jump to the next contour and continue the execution as if nothing have happened.

This problem can be solved by just increasing the dimension of x,y vectors.

2.6 Common Areas

This program has many common areas shared by different subprograms. The table IX illustrates the relation between the subprogramns through these common areas.

2.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

2.8 Usage

The 11 subprograms that compose this program are grouped in a single module called THE8M.FOR. Thus, these are the following procedures to work with this piece of code.

```
Compilation: $FOR  THE8M
Linking    : $LINK THE8M
Running    : $RUN  THE8M
```

3. Program Name: THE8KNT.FOR

3.1 Objective

Given a set of coordinates (x,y) of the contour and the character matrix map generated by the program THE8M.FOR, this program minimizes the number of knots in each contour by selecting automatically the break points according to the Lagrange's minimization approach, i.e.:

- Minimize the second derivative (maximize the smoothness) under the constraint of the maximum closeness error measurement less than a factor "S" which is a function of the number of breakpoints. For more details see Chapter III.
- The three-dimensional effect is carried out by setting the weighting function W(I) proportional to the density of contours in the neighborhood of each contour point (x,y). These measurements are done by overlaping the contours record over the character matrix map which are shown in Figure A.3.

3.2 Input Data

File FOR011.DAT: record containing the reference and incremental weights, W0 and ΔW respectively.

File FOR018.DAT: character matrix map containing the (connectivity four) contours, represented by the letters: A,B,.... Each letter defines a specific layer and the background is represented by the blank character.

File FOR020.DAT: record containing the (x,y) coordinates of the contours.

3.3 Output Data

File FOR022.DAT: record containing the (x,y) coordinates of the selected knots of each contour.

3.4 Program structure

This program consits of 13 subprograms organized according to three abstract levels of hierarchy shown in Figure A.3. At the first level there is a main program which handles the input/output functions. At the second level there are four subprograms which perform the following functions:

- Subprogram DPARAM: selection of the knots for the opened contours.

- Subprogram DPERCLO: selection of the knots for the closed contours.

- Subprogram TRIDI: introduction of the weighting factor to the contours points according to the density of contours in the neighborhood of that area being analysed.

- Subprogram DERIVATIVE: evaluation of the parametric functions "x" and "y" in the selected knots accomplished by the former subprograms.

The third level programs perform fundamental functions which descriptions are imbeded in the body of the source listing in Appendix C.

3.5 Constraints and Assumptions

The maximum number of coordinate pairs allowed per each contour is 4300. The character map must be a two-dimensional array with 256 rows and columns.

The following parameters were set for this experiment and they are assumed to be the default unless an explicit change is made.

- WINDOW = 5: It gives the span of the algorithm that measures the density in the neighborhood of each point belonging to the contour.

- K = 3: It gives the degree of the B-spline; in this case they are cubic splines.

- IPAR = 0: It gives the option for automatic parametrization. This means that the parameter to represent each curve (contour) is to be constructed according to equation 3.32. The reason to choose this procedure is to enforce the robustness of the overall process, i.e., keep it as general as possible.

- TOTPT = 0: This switch sets the program to a single pass mode.

- NEST = 1000: It gives the maximum number of knots allowed to be selected per each contour, what has demonstrated enough for our experiments.

- TOL = 0.001: It is the requested relative accuracy for the root of F(p) =S.

- MAXIT = 20: It is the maximum number of iterations allowed in the estimation of the root of F(p) = S.

3.6 Common Areas

There are two common areas. The first one is named CIDA and it is used to share data between the main program and the subroutine TRIDI. The second common area is named OPT1 and it is mentioned in the subroutines DPARAM and DPERCLO, but it is not used in our application here.

3.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

3.8 Usage

The 13 subprograms of this set are grouped into four modules in the following fashion:

MODULE	SUBPROGRAMS
THE8KNT	THE8KNT (main)
TRIDI	TRIDI
DPERCLO	DPERCLO, PBACK
DPARAM	DPARAM, BSPLN, COSSIN, ROTATE, BACK, KNOT, DISCO, RATION, DERIV

Each module is compiled individually. The linking and running are performed in the following way:

```
$LINK  THE8KNT, TRIDI, DPERCLO, DPARAM
$RUN   THE8KNT
```

4. Program Name: FILTREX.FOR

4.1 Objective

This program reads the file which contains the selected knots of the contours and eliminates the repeated points (knots) which appear due to the round off imposed in the program THE8KNT.FOR during the storage of the selected pairs (x,y).

Note: This routine eliminates the repetition pairs only inside each contour set not among different contours. This task is accomplished in the next step.

4.2 Input Data

File FOR0IR.DAT: (usually IR = 22) with (x,y) coordinates of the selected knots (which may be repeated due to round off). The reason for truncation is that the resolution of the grid is 1/256 in a zero to 255 integer range.

4.3 Output Data

File FOROIW.DAT: (usually IW = 23) with (x,y) coordinates of the selected knots without repetition in the same contour.

4.4 Program Structure

Only the main program with no subroutines or function call.

4.5 Constraints and Assumptions

The input records of selected knots are allowed to have at maximum 1000 elements. In order to change this constraint it is necessary to change the dimension statement.

4.6 Common Areas

Not applicable in this case.

4.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

4.8 Usage

```
Compilation: $FOR  FILTREX
Linking    : $LINK FILTREX
Running    : $RUN  FILTREX
```

5. Program Name: FRA5M.FOR

5.1 Objective

This program reconstructs the image as an uniform grid of pixels with the same size as the original and presumably with as lower distortion as possible. The set of the scattered points (compressed data) is fed to an algorithm based on the triangulation that was discussed in Chapter IV.

5.2 Input Data

File FOR019.DAT: number of contours in each level and the gray level value of that respective level.

File FOR023.DAT: (x,y) coordinates of the selected knots.

5.3 Output Data

File FOR031.DAT: uniform (256x256) grid representing the reconstructed image picture.

5.4 Program Structure

This program consists of seven subprograms distributed in four layers. The Figure A.4 shows the abstract hierarchical diagram. The description of each subprogram is provided in the source listing in the Appendix C.

5.5 Constraints and Assumptions

The maximum allowed number of input coordinate pairs (x,y) is 10000.

5.6 Common Areas

There is only one common area namely IDLC, which is shared by the subprograms PWLOT and IDLCTN.

5.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

5.8 Usage

The seven subprograms are grouped into two modules which are organized in the following manner:

MODULE	SUBPROGRAM
FRA5M	FRA5M
PWLM1	PWLOI, IDTANG, IDLCTN, PWLIN, EXTR, IDXCHG

Each module is compiled independently and the next steps are:

Linking: $LINK FRA5M, PWLM1

Running: $RUN FRA5M

6. Program Name: FRA4M.FOR

6.1 Objective

Analogous to the former program FRA5M.FOR, this program also reconstructs the uniform grid from a set of scattered points (selected knots). The difference here is regarding to the algorithm used to accomplish this task. Now a thin plate spline method is used to generate a C^1 surface, opposed to the former one that generates a C^0 surface. The description of this algorithm is in Chapter IV.

6.2 Input Data

File FOR012.DAT: parameter "NPPR" that is the short for number of points per region. This parameter controls the mesh of the auxiliary grid over which the thin plates splines are applied in order to reconstruct the image.

File FOR019.DAT: number of contours in each level and the gray level value of the respective level.

File FOR023.DAT: coordinate pairs (x,y) of the selected knots or pixels.

6.3 Output Data

File FOR031.DAT: uniform (256x256) grid representing reconstructed image picture.

6.4 Program Structure

This program consists of nine subprograms organized in five levels. The Figure A.5 shows the abstract hierarchical structure of the program tree. The description of each subprogram can be found in Appendix C.

6.5 Constraints and Assumptions

The maximum number of input coordinate pairs (x,y) is 15000. The mode switch was set to one, MODE = 1, that means: compute the coefficients for the local approximations by thin plate spline and return the grid of interpolated function values nx0, x0, ny0, y0, and f0 where:

```
nx0 = 256 : number of rows
 x0       : row coordinate
ny0 = 256 : number of columns
 y0       : column coordinate
 f0       : gray level value of coordinate pair (x0,y0)
```

6.6 Common Areas

Not applicable in this case.

6.7 Error Message

Embedded in the body of the respective code listed in Appendix C.

6.8 Usage

The nine subprograms are grouped into two modules which are organized in the following manner:

MODULE	SUBPROGRAM
FRA4M	FRA4M
F24	LOPTS, EVLPTS, GRID
	LOCLIP, CLOTPS, VSRTA,
	DECOMP, SOLVE

Each module is compiled independently and the next steps are:

Linking: $LINK FRA4M, F24

Running: $RUN FRA4M

7. Program Name: DISP3M.FOR

7.1 Objective

This program reads the (256x256) uniform grid (image) from unit "IR" (usually IR = 31) and converts it into a COMTAL's format picture, i.e., a 512x512 grid of pixels. In another words, an expansion two by two is carried out.

7.2 Input Data

File FOR0IR.DAT: (usually IR = 31) consisting of the uniform square grid (256x256) representing the image in a gray level range from zero to 256.

7.3 Output Data

File RITA.DAT: consisting of 512 records and 512 byte/record suitable to be displayed in COMTAL.

7.4 Porgram Structure

Only the main program with no subroutines or functions call.

7.5 Constraints and Assumptions

As we are dealing with pixels from the range zero to 255 and due to the fact that Fortran uses a two's complement form, each byte in Fortran covers the interval -128 to 127. Therefore, in order to represent a range from zero to 255 some transformation is accomplished in the following way: let "N" be the gray level for a particular pixel and "MB" be the actual element to be sent to the COMTAL device. The following algorithm will perform the necessary mapping:

```
IF (N ≤ 127) THEN
   MB = N
ELSE
   MB = -256 + N
END IF
```

This procedure will guarantee that each pixel value "N" will be seen by the COMTAL as an integer ranging from zero to 255.

7.6 Common Areas

Not applicable in this case.

7.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

7.8 Usage

Compilation : $FOR DISP3M

Linking : $LINK DISP3M

Running : $RUN DISP3M

8. Program Name: RMSER

8.1 Objective

This program gives a quantitative measurement of the error between the original image (smooth version) and the reconstructed version from the compressed data. The metrics used are:

- Root mean square error (RMS)

$$E_{RMS} = SQRT \left[\sum_{i=1}^{N} \sum_{j=1}^{N} (F_{i,j} - \hat{F}_{i,j})^2 / N^2 \right]$$

- Relative error

$$E_{REL} = SQRT \left[\sum_{i=1}^{N} \sum_{j=1}^{N} (F_{i,j} - \hat{F}_{i,j})^2 / \sum_{i=1}^{N} \sum_{j=1}^{N} F_{i,j}^2 \right]$$

8.2 Input Data

File IMA31.DAT: consisting of the original image (256x256) grid of gray level value in the range zero to 255 to be used as reference.

File FOR0IR.DAT: (usually IR = 31) consisting of the uniform square grid (256x 256) of the reconstructed image to be evaluated.

8.3 Output Data

- Root means square error : E_{RMS}
- Relative error : E_{REL}

8.4 Program Structure

Only one block (main program).

8.5 Constraints and Assumptions

The input files IMAG1.DAT and FOR031.DAT are assumed to be arrays of integer with dimensions (256x256) with gray levels, i.e., pixel brightness value varying from zero to 255.

8.6 Common Areas

Not applicable in this case.

8.7 Error Messages

Embedded in the body of the respective code listed in Appendix C.

8.8 Usage

```
Compilation : $FOR  RMSER
Linking     : $LINK RMSER
Running     : $RUN  RMSER
```

A.2 FILES

As we can see from Figure A.1, the programs and files are the basic elements of our image procesing system. The files work like a glue that links the programs among themselves. In this section, the structure of the files and their contents are described. They are presented in a chronological fashion as they are generated and used according to the diagram in Figure A.1.

1. FILE TAD.DAT

It is the input picture file containing the original image to be compressed. It is in the format suitable to be

displayed at COMTAL (image displaying device). The expansion factor is four, resulting in a grid with (512x512) pixels. This file is the input of the program WDAT1.PAS and it is stored as a packed array.

2. FILE IMAG1.DAT

This file is the output of the program WDAT1.PAS and the first of the two inputs of program THE8M.FOR. It contains image file in its reduced form, without the expansion factor four. Each pixel is represented by an "INTEGER*4" type, in other words four bytes are required per each pixel. From now on, as we will be dealing with the Fortran programs, the Fortran convension will be the default. For example, in the Fortran environment the generic file FORxyz.DAT is by default assigned to the logical unit xyz where x, y, and z are digits from zero to nine.

When we say that a file has the format 2I5, this means that each record of this file stores two integer numbers in two consecutive fields of length five for each field, so that these numbers are right justified in their prespective fields.

3. FILE FOR010.DAT

It is the second input of the program THE8M.FOR. This is used to input two parameters: "NC", number of contour layers and "THRES", threshold value, so that the contours with less points than the threshold will be pruned. The format of the data in this file is 2I5.

4. FILE FOR011.DAT

It is an output of the program THE8KNT.FOR and it is used to input the two parameters W0, weight reference value

and ΔW, incremental weight. The format of the data in this file is 2F5.2.

5. FILE FOR012.DAT

This file is an input of the program FRA11.FOR, thin plate algorithm implementation. It is used to input the parameter "NPPR" that is an estimate of the number of points per region into which the whole image will be divided and the thin plates evaluated.

The format of this file is I2.

6. FILE FOR018.DAT

This file is the character mapping according to Figure 3.4, and it contains the contour of the given image such that the layer is represented by a string of characters over a background made up of blank characters. This file contains 512 records, each one with the format 128A1.

7. FILE FOR019.DAT

This file is the output of the program THE3M.FOR, and the input of the programs FOR4M.FOR and FOR5M1.FOR. It contains in each record the information regarding to one specific layer, namely the number of contours and the gray level value of such a layer. It will be used in the reconstruction process. The format of this file is I5, F10.5. A sentinel record (-3, -3.) is used to mark the end of the data stream.

8. FILE FOR020.DAT

This file is an output of the program THE8M.FOR and the input of the program THE8KNT. It contains the coordinate pairs (x,y) of each contour which are concatenated and

separated by a flag that gives the number of points in the following contour and its type (-1 if it is opened; -2 if it is closed.)

This file format is 2I5 and it is terminated by a sentinel (-3,-3).

9. FILE FOR022.DAT

This file is the output of the program THE3KNT.FOR and the input of the program FILTREX.FOR. It contains the selected knots (x,y) and its format is the same as FOR020.DAT, i.e., 2I5.

10. FILE FOR023.DAT

This file is the output of the program FILTREX.DAT and the input of the programs FOR4M.FOR and FORTM.FOR. It contains basically the same data as the file FOR022.DAT, except that the repeated pairs (x,y) in the same contour due to the truncation from floating point to integer type representation are filtered out.

11. FILE FOR031.DAT

This file is the output of the program FRA4M.FOR and FRA5M.FOR, and the input of the programs DISP3M.FOR and RMSER.FOR. It contains the reconstructed image, i.e., a grid of (256x256) integer values ranging from zero to 255, and its format is 32I4.

12. FILE RITA.DAT

This file is the output of the program DISP3M.FOR, and it is the reconstructed image suitable to be displayed at the COMTAL, where some qualitative measurements are done. This file is already in the expanded form, i.e., (512x512) grid.

In this Appendix A, we have seen the program environment and its two basic components: the programs and the files. More information about the discussed programs are provided in the Appendix C where a source listing of each program can be found.

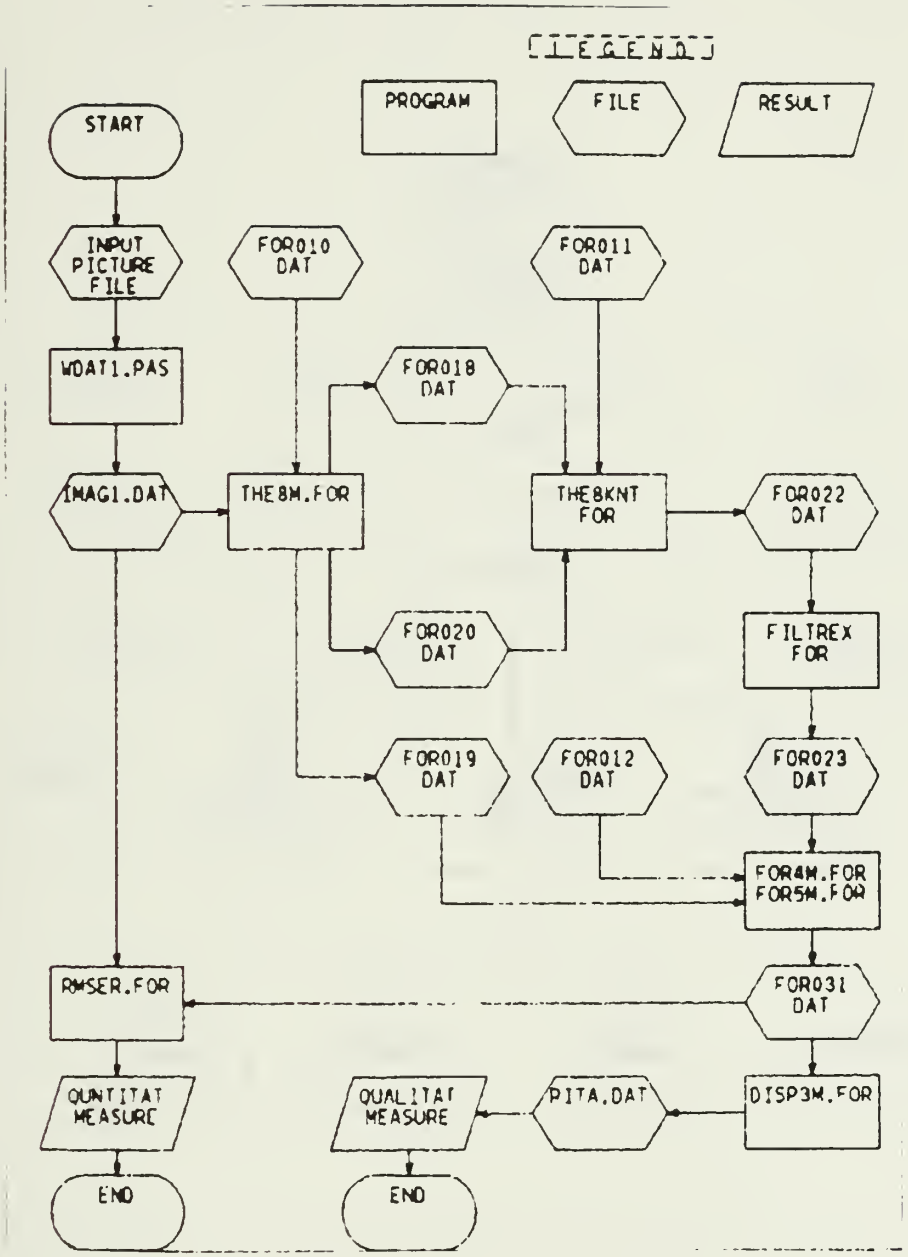

Figure A.1 Program Environment.

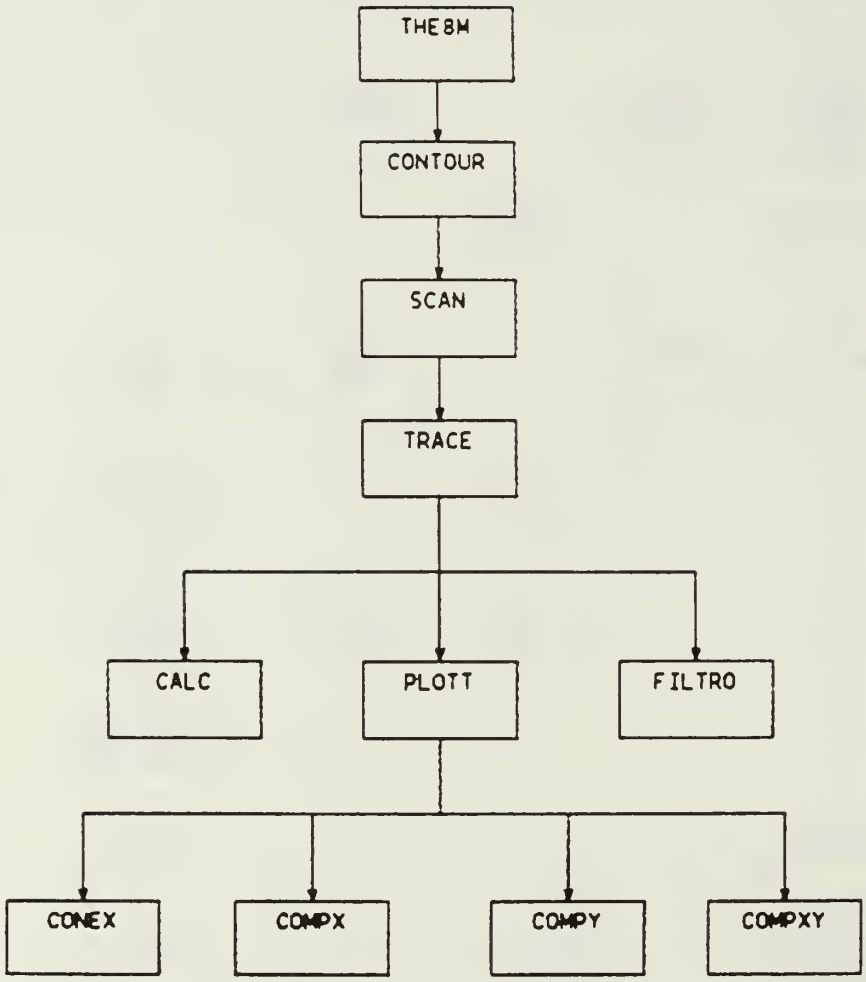

Figure A.2 Contour Generating - Program Structure.

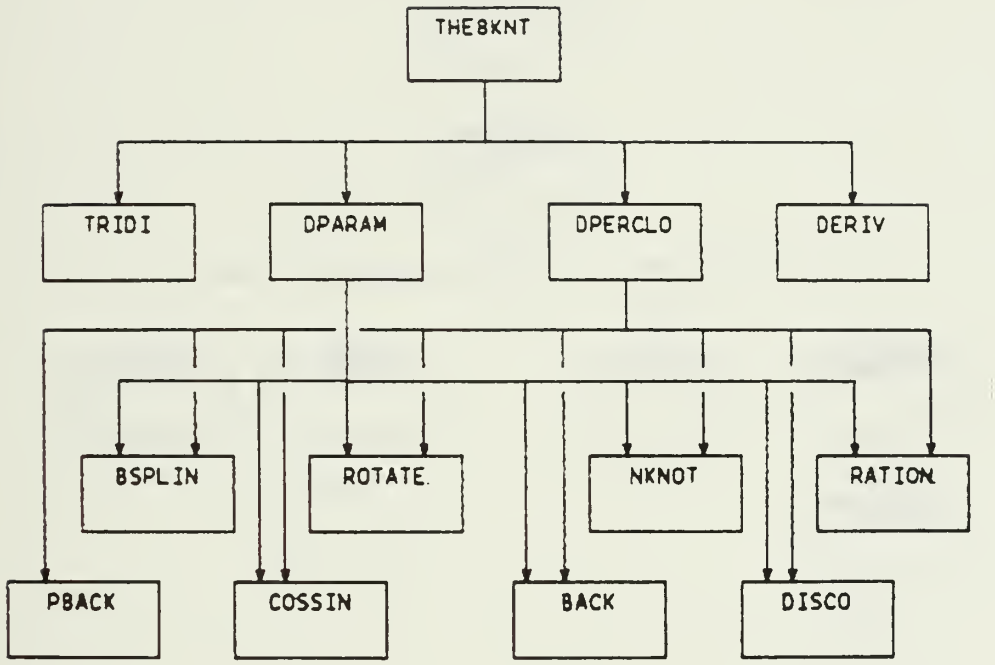

Figure A.3 Knot Selection - Program Structure.

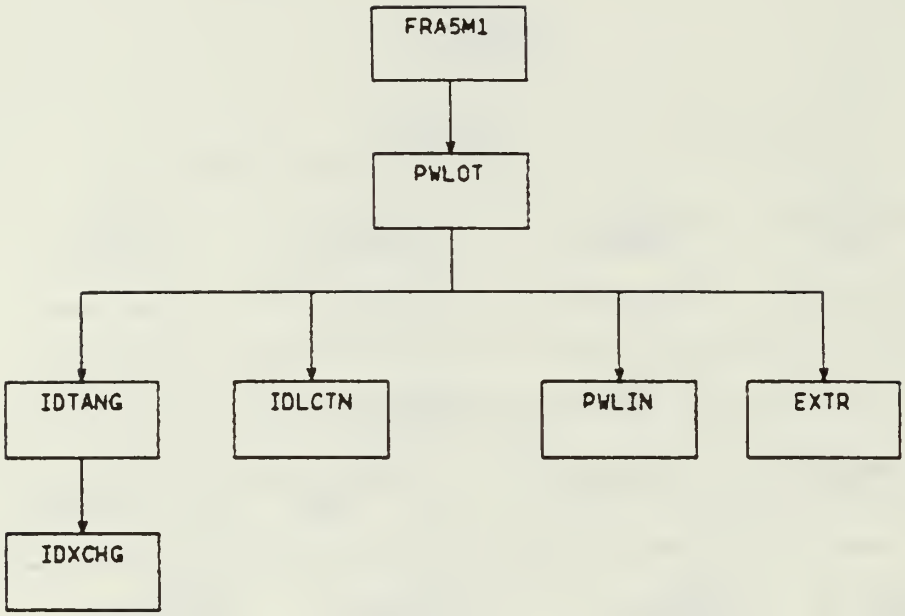

Figure A.4 Triangulation Method - Program Structure.

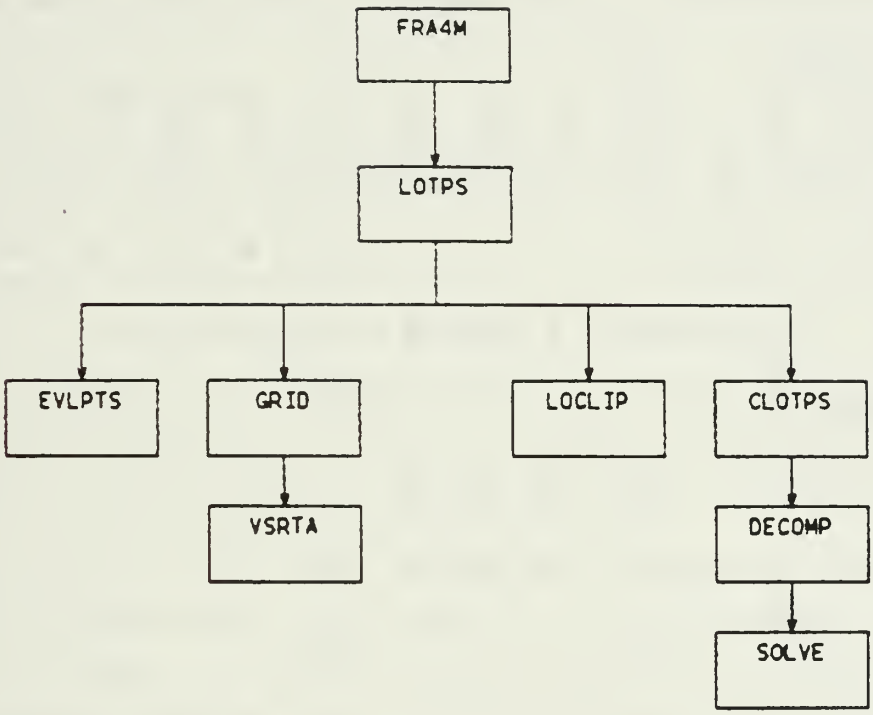

Figure A.5 Thin Plate Spline Method - Program Structure.

TABLE IX

Common Areas Cross References

COMMON AREAS \ SUBPROGRAMS	MAIN	CONTOUR	SCAN	TRACE	CALC	PLOTT	COMEX	COMPX	COMPY	COMPXY
TINA	●					●	●	●	●	●
ELI	●	●	●	●	●					
ISABEL	●					●				
DAYHOF		●	●	●	●					
INTFAC		●	●	●	●	●				
TABS		●				●				
FEFE		●				●	●	●	●	●
RITA		●				●				
DITS		●	●			●				
RIB						●	●	●	●	●

APPENDIX B
MORE ON COMPRESSION

The data compression scheme, as it was presented in the former chapters, is a two step procedure so that the first one is accomplished through the contour process and the second one by the application of the knot selection algorithms over the contours generated in the first step. Besides, it is clear to see that this coding process is irreversible. By this we mean that the original data can not be exactly reconstructed from the compressed set of data, rather it can be only approximated, and as it was enforced in previous chapters, one of the goals of the whole image processing system was to reduce the intrinsic error associated with the irreversible coding process.

It was also seen that there is a trade-off between the achieved compression ratio and the error between the reconstructed image and the original one, i.e., for a higher compression ratio we would expect higher error measurements and vice-versa.

There is another class of coding called reversible coding in which a set of given numbers is mapped into another set of numbers; the operation is error free and it has some important applications. A well known case of reversible code is the run length coding discussed by Gonzales [Ref. 1].

A close examination of the compressed data generated in the first and second phases shows that the stream of points representing the contour exhibits an interesting property, the coordinate pair (x,y) and its successor or predecessor differs by a small number.

The Figure B.1 shows a histogram of the maximum increments (x,y) for a typical contour diagram. As we can see, the concentration of small increments is evident.

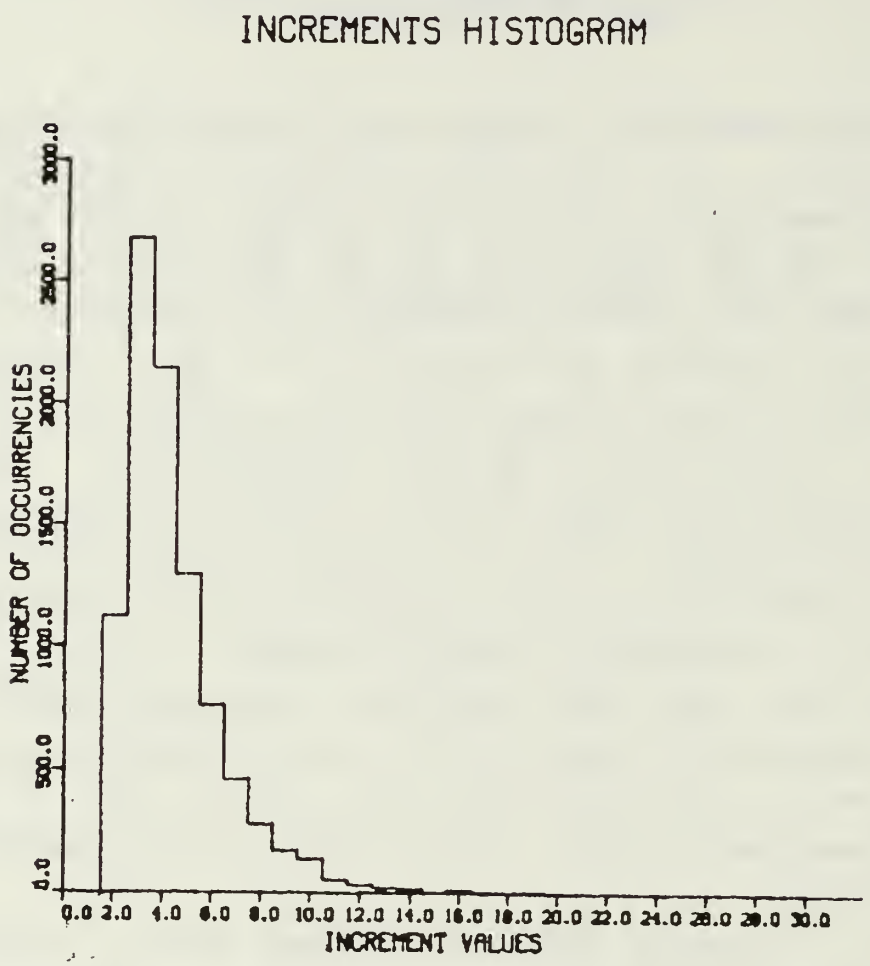

Figure B.1 Histogram for the Contours Increment.

We can explore this characteristic by increasing the compression ratio by a factor of about two in a reversible fashion with minimal effort. The process is somewhat similar to the run-length encoding and it works in the following way: instead of using two bytes to store each selected knot, a (x,y) pair of integer ranging from zero to 255, we store only the first knot of each contour (origin) and from there we store the increments between each point and its

predecessor. From the histogram in Figure B.1, we can see that the majority of increments between consecutive points lies in the range from minus seven to plus seven, excluding zero, because repeated points are not allowed in the contours (see function FILTREX in Appendix A). Therefore, we can use the value correspondent to the increment zero as a flag to indicate that one of the two coordinates or eventually both, have generated increment that has absolute value greater than seven.

For each contour the selected knots are packed in the following way:

- First point (origin) uses two bytes: the first one to store x coordinate and the second one to store y coordinate.
- From the second point and further on the following rule is applied: if the absolute value of the increment of x and y are both less than seven, use one nible to store each increment; if one of the coordinates or both have an absolute value greater than seven, than use the flag "zero" to indicate out of range followed by the value of increment of x stored in one byte and the increment of y stored in the next byte.

The Figure B.2 shows the original stream of data and the packed one.

As we have seen, the algorithm is simple, efficient, easy to implement, and it increases the data compression by a factor of about two without increasing the entropy of the system.

ORIGINAL RECORD OF CONTOURS

8	-2
70	31
72	33
76	33
77	35
80	32
83	32
73	32
70	31
10	-1

RECORD OF CONTOUR INCREMENTS

8	-2
70	31
2	2
4	0
1	2
3	-2
3	0
-10	0
-3	-1
10	-1

PACKED RECORD

8	-2
70	31
0010	0010
0100	0000
0001	0010
0011	1010
0011	0000
0000	0000
1000	1010
0000	0000
10	-1

ONE BYTE PER ROW

Figure B.2 Original and Packed Records of Contours Data.

APPENDIX C
SOURCE CODE LISTING

The Pascal and Fortran programs listed below were used in this work and are presented according to the following order.

- WDAT1 PASCAL
- THE8M FORTRAN
- THE8KNT FORTRAN
- FILTREX FORTRAN
- FRA5M FORTRAN
- FRA4M FORTRAN
- DISP3M FORTRAN
- RMSER FORTRAN

```
PROGRAM WDAT1 (IFILE,INPUT,OUTPUT);

(* THIS IS A PASCAL PROGRAM THAT READS A PACKED FILE CONTAINING
   THE IMAGE DATA. THE IMAGE IS REPRESENTED BY A GRID OF 256X256
   PIXELS  IN THE RANGE FROM "0" TO 256. THE OUTPUT OF THIS PROGRAM
   IS AN UNPACKED FILE SUITABLE TO BE READ BY A FORTRAN PROGRAM.*)

TYPE
   BYTE=0..255;
   IDATA = PACKED ARRAY _1..512' OF BYTE;
   ODATA = ARRAY_1..256,1..256' OF BYTE;
   OUTPUTDAT=ARRAY_1..256' OF BYTE;

VAR
   IFILE: FILE OF IDATA;
   MYDATA: ODATA;
   COLCOUNT: 1..256;
   L,J,I,K :1..512;
   OFILE: FILE OF OUTPUTDAT;

BEGIN
   OPEN(IFILE,'MYFILE.DAT',HISTORY:=OLD,RECORD_LENGTH:=512);
   OPEN(OFILE,'IMAGI.DAT',HISTORY:=NEW,ACCESS_METHOD:=SEQUENTIAL,
      RECORD_TYPE:=FIXED);
   RESET(IFILE);
   REWRITE(OFILE);
   WRITELN('START READING THE FILE MYDATA.DAT');
   IF EOF(IFILE) THEN
     BEGIN
       WRITELN(OUTPUT);
       WRITELN('NO DATA FOR IFILE');
     END
   ELSE
       WRITELN('YES, THERE IS DATA');
   FOR I:=1 TO 256 DO
     BEGIN
       GET(IFILE);
       FOR COLCOUNT:=1 TO 256 DO
          BEGIN
            K:=COLCOUNT*2 - 1;
            MYDATA_I,COLCOUNT':=IFILE;_K';
            OFILE;_COLCOUNT':=MYDATA_I,COLCOUNT';
          END;
          PUT(OFILE);
          GET (IFILE);
     END
END.
```

```
C PROGRAM THEEM.FOR
C
C CESAR 09/MAY/83
C
C THIS IS A FORTRAN PROGRAM THAT GENERATES THE CONTOURS FROM A GIVEN
C IMAGE REPRESENTED BY A UNIFORM GRID. THE CORE OF THIS PROGRAM IS
C THE SUBROUTINE "CONTUR" FROM THE NONIMSL LIBRARY WITH SOME MODIFICATIONS.
C
C
C INPUT PARAMETERS
C       FILE IMAG1.DAT: GIVEN IMAGE TO BE CONTOURED.
C       FILE FOR10.DAT: FILE CONTAINING THE TWO PARAMETERS: NC AND THRES
C       WHERE NC IS THE NUMBER OF LAYERS AND THRES IS THE THRESHOLD FACTOR.
C
C OUTPUT PARAMETERS
C    FILE FOR18.DAT: FILE CONTAINING THE FULLY CONNECTED MAP OF CONTOURS
C       PROJECTED IN THE SPATIAL COORDINATES PLANE, SO THAT EACH LAYER IS
C       REPRESENTED BY A LETTER SARTING WITH "A" FOR THE LOWEST LEVEL.
C       FILE FOR19.DAT: FILE CONTAINING THE GRAY LEVEL OF THE LAYERS IN
C       WHICH THE IMAGE WAS SECTIONED AND THE NUMBER OF CONTOURS IN EACH
C       LEVEL.
C       FILE FOR20.DAT: FILE CONTAINING THE (X,Y) PAIRS FOR EACH CONTOUR.
C

C VARIABLE DECLARATION
C
      INTEGER BL
      COMMON/TINA/IMAT(0:255,0:255)
      DIMENSION F(256,256),CL(20),ITA(256)
      INTEGER IM,JM,IMAX,IPLOT,JPLOT
      REAL*8 TITLE(12)
      LOGICAL*1 LTG(3)
      COMMON/ELI/F
      INTEGER NC,THRES
      COMMON/ISABEL/THRES
C
C INITIAL PARAMETERS
      DATA BL/' '/
      DATA IM,JM,IMAX,IPLOT,JPLOT/256,256,256,255,255/
      TITLE(1)='AAA'
      LTG(1)=.FALSE.
      LTG(2)= .FALSE.
      LTG(3)=.FALSE.
C
C CREATE A 256X256 BLANK  MATRIX "BACKGRACUND" FOR FILE FOR18.DAT.

      DO 60 I=0,255
         DO 56 J=0,255
```

```
              IMAT(I,J)=BL
 50         CONTINUE
 60     CONTINUE
C
C CONSTRUCTION OF FUNCTION: READ THE GIVEN IMAGE FROM THE FILE
C IMAG1.DAT
C
       OPEN(UNIT=7,NAME='IMAG1.DAT',RECORDSIZE=256,ACCESS='DIRECT',
      1 TYPE='UNKNOWN')
       DO I=1,256
C        TYPE *,'RECORD # ',I
         READ(7'I) ITA
         DO J=1,256
           F(I,J)=FLOAT(ITA(J))
         END DO
       END DO
       TYPE *,'ALL RECORDS WERE READ'
C
C
C SMOOTHING OF DATA
       TYPE *,'STARTING OF SMOOTHING'
       DO I=1,256
         SAVE=F(I,2)
         F(I,2)=(F(I,1)+F(I,2)+F(I,3))/3.
         DO J=3,255
           ACT=F(I,J)
           F(I,J)=(SAVE+ACT+F(I,J+1))/3.
           SAVE=ACT
         END DO
       END DO
       DO J=1,256
          SAVE=F(2,J)
          F(2,J)=(F(1,J)+F(2,J)+F(3,J))/3.
          DO I=3,255
            ACT=F(I,J)
            F(I,J)=(SAVE+ACT+F(I+1,J))/3.
            SAVE=ACT
         END DO
       END DO
       TYPE *,'END SMOOTHING'
C
C INVOKE THE CONTOURING ROUTINE
C
       CALL CONTUR(IM,JM,IMAX,CL,NC,TITLE,IPLOT,JPLOT,LTG)
C
C STORE THE CHARACTER MATRIX OF CONTOUR IN A FILE FOR018.DAT
       DO I=0,255
          WRITE(18,126)(IMAT(I,J),J=0,127)
```

```
          WRITE(18,126)(IMAT(I,J),J=128,255)
        END DO
        STOP
  126   FORMAT(128A1)
  140   FORMAT(2I5)
        END
C
C
C*******************************************************************
C
      SUBROUTINE CONTUR(M,N,MX,CL,NL,TITLE,IW,IH,LTG)
C
C     SUBROUTINE CONTUR    (VERSATEC SOFTWARE VERSION)        (J5)
C
C     1. IDENTIFICATION
C
C        A. NAME: GENERATE CONTOUR GRAPH (CONTUR) - J5
C
      REAL*8 TITLE(12)
      REAL*8 WIDTH/'WIDTH'/,HEIGHT/'HEIGHT'/,WHICH
      DIMENSION CL(1)
      COMMON/ELIMAM(256,256)
      DIMENSION           REC(1900),  X(4800), Y(4800)
      DIMENSION IPT(3,3),INX(8),INY(8)
      COMMON /DAYHOF/ MT,NT,NI,IX,IY,IDX,IDY,ISS,IT,IV,NP,NQ,JT,
     1                PY,REC,CV,    IPT,INX,INY,DL,RA,THE
      COMMON /INTFAC/ X,Y
      DIMENSION DITSX(7),DITSY(7)
      LOGICAL*1 LTG(1),MINUS,LABL,MP(2),MPA(2)
      COMMON/TABL/ TABC(20,6),JC
      COMMON/DITS/XMIN,YMIN,SLOPEX,SLOPEY,DITSDX,DITSDY,IDIR,LABL,MINUS
      COMMON/FEFE/LCHAR
      INTEGER CONTVE(20),IXSENT,IYSENT,NSAME
      COMMON/FITA/NSAME
C
      MP(1)=.FALSE.
      MP(2)=.FALSE.
      GO TO 2945
      ENTRY CONTR1  (M,N,MX,CL,NL,TITLE,IW,IH,LTG,MPA)
      MP(1)=MPA(1)
      MP(2)=MPA(2)
 2945 JC=0
      LABL=LTG(1)
C     CHECK IW PARAMETER
      WHICH=IW
      IF(IW) 1,1,2
    1 WRITE(6,60) WHICH
   60 FORMAT('0',T7,A8,' OF CONTOUR GRAPH ILLEGAL.')
```

```
   71 WRITE(6,64)
   64 FORMAT('0',T7,'NO GRAPH WILL BE PRODUCED.')
      RETURN
C     CHECK IF IW IS TOO WIDE
2     IF (IW-255) 3,3,40
   40 WRITE(6,61)
61    FORMAT('0',T7,'IW PARAMETER GREATER THAN 255.  CONTUR WILL SET IW=
     *255')
      IW=255
C     NOW CHECK IH PARAMETER
    3 IF(IH) 4,4,5
    4 WHICH=IH
      GO TO 1
    5 DITSDX=(N-1.0)/IW
      DITSDY=(-1.0+M)/IH
      W=IW
      H=IH
      XMIN=1.0
      YMIN=-M
      SLOPEX=1.0/DITSDX
      SLOPEY=1.0/DITSDY
      DITSX(1)=1.0
      DITSX(4)=1.0
      DITSX(5)=1.0
      DITSX(2)=N
      DITSX(3)=N
      DITSY(1)=-1.0
      DITSY(2)=-1.0
      DITSY(5)=-1.0
      DITSY(3)=-M
      DITSY(4)=-M
      DO 2011 I=1,5
      DITSX(I)=SLOPEX*(DITSX(I)-XMIN)
 2011 DITSY(I)=SLOPEY*(DITSY(I)-YMIN)
      IF(MP(1)) GO TO 2012
      STARTP=(21.0-IW)/2.0
C      CALL PLOTS(0,0,0)
C      CALL PLOT(2.,STARTP,-3)
C 2012 CALL SCALE(DITSX,W,5,1)
C      CALL SCALE(DITSY,H,5,1)
C      CALL LINE(DITSY,DITSX,5,1,0,0)
2012  DITSX(1)=DITSX(1)-.5
      DITSX(5)=DITSX(1)
      DITSX(4)=DITSX(4)-.5
      DITSX(2)=DITSX(2)+.5
      DITSX(3)=DITSX(3)+.5
      DITSY(1)=DITSY(1)+.5
      DITSY(5)=DITSY(1)
```

```
      DITSY(2)=DITSY(2)+.5
      DITSY(3)=DITSY(3)-.5
      DITSY(4)=DITSY(4)-.5
C      CALL LINE(DITSY,DITSX,5,1,0,0)
      SLOPEX=1.0/DITSDX
      SLOPEY=1.0/DITSDY
      IENDX=IW+1
      IENDY=IH+1
      IF(.NOT.LTG(2)) GO TO 34
C     DRAW TIC MARKS ON OUTER FRAME
C     START ON LEFT EDGE GOING DOWNWARD
      IFLAG=0
      ZINGX=-.1
      ZINGY=0.0
      ZX=0.0
      ZY=-1.0
      CX=DITSX(1)
      CY=DITSY(1)-.5
      IEND=IENDY
 2222 IFLAG=IFLAG+1
      DO 2022 I=1,IEND
C      CALL PLOT(CY,CX,3)
      COORDX=CX+ZINGX
      COORDY=CY+ZINGY
C      CALL PLOT(COORDY,COORDX,2)
      CX=CX+ZX
 2022 CY=CY+ZY
      GO TO (21,22,23,24),IFLAG
C     NOW DO THE RIGHT EDGE GOING DOWNWARD
   21 ZINGX=.1
      CX=DITSX(2)
      CY=DITSY(2)-.5
      GO TO 2222
C     NOW DO TOP EDGE
   22 ZINGX=0.0
      ZINGY=.1
      ZX=1.0
      ZY=0.0
      CX=DITSX(1)+.5
      CY=DITSY(1)
      IEND=IENDX
      GO TO 2222
C     NOW DO THE BOTTOM EDGE
   23 ZINGY=-.1
      CX=DITSX(4)+.5
      CY=DITSY(4)
      ZINGY=-.1
      GO TO 2222
```

```
C       NOW LABEL TIC MARKS
C       DO X-DIRECTION FIRST, TOP EDGE
C       POSITION PEN
   24   DELTAX=DITSDX
        IFLAG=0
        ZX=1.0
        ZY=0.0
        CX=DITSX(4)+.35
        CY=DITSY(4)-.15
 3033   IFLAG=IFLAG+1
        XZERO=1.0
        DO 3333 I=1,IEND
C        CALL NUMBER(CY,CX,.07,XZERO,90.,1)
        CX=CX+ZX
        CY=CY+ZY
 3333   XZERO=XZERO+DELTAX
        GO TO (31,32,33,34),IFLAG
C       LABEL BOTTOM EDGE TIC MARKS
   31   CX=DITSX(1)+.35
        CY=DITSY(1)+.2
        GO TO 3033
C       LABEL LEFT EDGE OF TIC MARKS
   32   CX=DITSX(4)-.4
        CY=DITSY(4)+.54
        DELTAX=DITSDY
        IEND=IENDY
        ZX=0.0
        ZY= 1.0
        GO TO 3033
C       NOW LABEL RIGHT EDGE TIC MARKS
   33   CX=DITSX(3)+.15
        CY=DITSY(3)+.54
        GO TO 3033
C       CHECK IF GRID DESIRED
   34   IF(.NOT.LTG(3)) GO TO 35
C       DRAW INCH BY INCH GRID
        DATA LMASK/ZFFFF/
C        CALL GRID(.0,.0,IH,1.,IW,1.,LMASK)
   35   CONTINUE
        NLEV=NL
C       CHECK IF CONTUR IS TO  COMPUTE LEVELS
        IF(NLEV) 6,7,11
C       NL NEGATIVE  MEANS CONTOUR MUST COMPUTE LEVELS
C       SCAN MATRIX FOR LOWEST AND HIGHEST LEVELS
    6   NLEV=-NLEV
        FMIN=A(1,1)
        FMAX=FMIN
        DO 9 I=1,M
```

```
      DO 9 J=1,N
      IF(AM(I,J).LT.FMIN) FMIN=AM(I,J)
      IF(AM(I,J).GT.FMAX) FMAX=AM(I,J)
    9 CONTINUE
C     CHECK IF MIN AND MAX ARE EQUAL
      IF(FMIN.GE.FMAX) GO TO 300
      DELTAL=(FMAX-FMIN)/(NLEV-1)
C     SET UP CL ARRAY
      DO  I=1,NLEV
        CL(I)=FMIN+(I-1)*DELTAL
      END DO
   11 DO  I=1,NLEV
        LCHAR=I
        TYPE *,'CALL CONTOUR #',I
        NSAME = 0
        CALL SCAN(M,N,MX,CL(I))
        CONTVE(I)=NSAME
        TYPE *,'NUMBER OF IDENTICAL CONTOURS',NSAME
        WRITE(19,125) CONTVE(I),CL(I)
      END DO
      IXSENT=-3
      IYSENT=-3
      WRITE(20,124) IXSENT,IYSENT
      RSENT= -3.
      WRITE(19,125)IXSENT,RSENT
      IF(.NOT.LABL) GO TO 778
      IF(JC.EQ.0) GO TO 778
      DO 777 I=1,JC
      COORDX=TABC(I,4)
      COORDY=TABC(I,5)
      CLEV=TABC(I,6)
C      CALL NUMBER(COORDY,COORDX,.07,CLEV,90.,1)
  777 CONTINUE
778    CONTINUE
C778   CALL SYMBOL(-1.0,0.0    ,.21,TITLE(7),90.,48)
C      CALL SYMBOL(-1.5,0.0    ,.21,TITLE(1),90.,48)
C      IF (.NOT.MP(2)) CALL PLOT(0.,0.,+999)
C      WRITE(6,779) TITLE
C  779 FORMAT('0CONTUR TITLED'//1X,6A8/1X,6A8/'0HAS BEEN PLOTTED'//)
      RETURN
  300 WRITE(6,62) FMIN
   62 FORMAT('0',T7,'DATA MATRIX HAS ONLY ONE LEVEL=',E15.3)
      GO TO 71
    7 WRITE(6,63)
   63 FORMAT('0',T7,'NUMBER OF LEVELS REQUESTED=0.')
      GO TO 71
  124 FORMAT(2I5)
  125 FORMAT(I5,F10.5)
```

```
      END
C
C*******************************************************************************
C
C  SUBROUTINE SCAN
C
      SUBROUTINE SCAN(M,N,MX,CL)
C THIS PROGRAM IS WRITTEN BY M.O.DAYHOFF
      DIMENSION REC(1900),  X(4800), Y(4800)
      COMMON/ELIZAM(256,256)
      DIMENSION IPT(3,3),INX(8),INY(8)
      COMMON /DAYHOF/ MT,NT,NI,IX,IY,IDX,IDY,ISS,IT,IV,NP,NQ,JT,
     1                PY,REC,CV,     IPT,INX,INY,DL,RA,THE
      COMMON /INTFAC/ X,Y
      LOGICAL *1 LABL,MINUS
      COMMON/CITS/XMIN,YMIN,SLOPEX,SLOPEY,CITSDX,CITSDY,IDIR,LABL,MINUS
      D=0.
      R=1.
      TH = 1.570796
      NP=0
      DL=D
      RA=R
      THE=TH
      MT=N
      NT=M
      CV=CL
      IF(IZW-120631) 1,3,1
    1 IPT(1,1)=8
      IPT(1,2)=1
      IPT(1,3)=2
      IPT(2,1)=7
      IPT(2,3)=3
      IPT(3,1)=6
      IPT(3,2)=5
      IPT(3,3)=4
      INX(1)=-1
      INX(2)=-1
      INX(3)=0
      INX(4)=1
      INX(5)=1
      INX(6)=1
      INX(7)=0
      INX(8)=-1
      INY(1)=0
      INY(2)=1
      INY(3)=+1
      INY(4)=+1
      INY(5)=0
```

```
      INY(6)=-1
      INY(7)=-1
      INY(8)=-1
      IZW=120631
    3 XT=MT
      DO 58 J=1,1900
   58 REC(J)=C
      ISS=0
    2 MT1=MT-1
      IDIR=1
      DO 110 I=1,MT1
      IF(AM(1,I)-CV) 55,110,110
   55 IF(AM(1,I+1)-CV) 110,57,57
   57 IX=I+1
      IY=1
      IDX=-1
      IDY=0
      CALL TRACE
  110 CONTINUE
      NT1=NT-1
      IDIR=2
      DO 20 I=1,NT1
      IF(AM(I,MT)-CV) 15,20,20
   15 IF(AM(I+1,MT)-CV) 20,17,17
   17 IX=MT
      IY=I+1
      IDX=0
      IDY=-1
      CALL TRACE
   20 CONTINUE
      IDIR=3
   22 DO 30 I=1,MT1
      MT2=MT+1-I
      IF(AM(NT,MT2)-CV) 25,30,30
   25 IF(AM(NT,MT2-1)-CV) 30,27,27
   27 IX=MT2-1
      IY=NT
      IDX=1
      IDY=0
      CALL TRACE
   30 CONTINUE
      IDIR=4
      DO 40 I=1,NT1
      NT2=NT+1-I
      IF(AM(NT2,1)-CV) 35,40,40
   35 IF(AM(NT2-1,1)-CV) 40,37,37
   37 IX=1
      IY=NT2-1
```

```
      IDX=0
      IDY=1
      CALL TRACE
   40 CONTINUE
      IDIR=5
      ISS=1
      NT1=NT-1
      MT1=MT-1
      DO 10 J=2,NT1
      DO 10 I=1,MT1
      IF(AM(J,I)-CV)5,10,10
    5 IF(AM(J,I+1)-CV) 10,7,7
    7 COM=100*(I+1)+J
      IF (NP) 12,11,12
   12 DO 9 ID= 1,NP
      IF (REC(ID)-COM) 9,10,9
    9 CONTINUE
   11 IX= I+1
      IY=J
      IDX=-1
      IDY=0
      CALL TRACE
   10 CONTINUE
      RETURN
      END
C
C*****************************************************************************
C
      SUBROUTINE TRACE
      DIMENSION REC(1900),  X(4800), Y(4800)
      COMMON/ELI/AM(256,256)
      DIMENSION IPT(3,3), INX(8),INY(8)
      COMMON /DAYHOF/ MT,NT,NI,IX,IY,IDX,IDY,ISS,IT,IV,NP,N ,JT,
     1                PY,REC,CV,     IPT,INX,INY,DL,RA,THE
      COMMON /INTFAC/ X,Y
      PY=0.0
      RC=  COS (THE)*RA
      RS=  SIN (THE)*RA
  501 JT=0
      N=0
      IX0=IX
      IY0=IY
      ISX=IDX+2
      ISY=IDY+2
      IS=IPT(ISX,ISY)
      JTB=0
      IS0=IS
      IF(IS0-8)18,18,17
```

```
   17 ISO=ISO-8
   18 IT=0
    5 CONTINUE
      CALL CALC
      NZ=N
      N=NZ
      IF (IT*JT-1) 49,49,47
   47 XS=X(N-1)
      YS=Y(N-1)
      X(N-1)=X(N)
      Y(N-1)=Y(N)
      X(N)=XS
      Y(N)=YS
   49 IS=IS+1
      JT=IT
    9 IF (IS-9) 8,7,7
    7 IS=IS-8
    8 IDX=INX(IS)
      IDY=INY(IS)
      IX2=IX+IDX
      IY2=IY+IDY
      JTB=JTB+1
      IF (JTB-4797) 51,51,308
  308 PRINT                    103,CV,X(N),Y(N)
  103 FORMAT(1HO,23HA CONTOUR LINE AT LEVEL,E12.5,21H WAS TERMINATED AT
     1X=,E12.5,3H Y=,E12.5/
     2 48H BECAUSE IT CONTAINED MORE THAN 4799 PLOT POINTS  )
      RETURN
C
C    SHOULD TEST HERE FOR MAXIMUM NUMBER OF PLOTTABLE POINTS IN SEGMENT.
   51 CONTINUE
      IF (ISS) 10,10,20
   20 IF(IX-IXO) 12,21,12
   21 IF(IY-IYO) 12,22,12
   22 IF(IS-ISO) 12,23,12
   23 CONTINUE
      CALL CALC
      GO TO 73
   10 IF(IX2) 13,50,13
   13 IF (IX2-MT) 19,19,50
   19 IF (IY2) 11,50,11
   11 IF (IY2-NT) 12,12,50
   12 IF(CV-AM(IY2,IX2)) 206,206,5
  206 IF (IDX**2+IDY**2-1) 213,6,213
  213 DCP=(AM(IY,IX)+AM(IY,IX2)+AM(IY2,IX)+AM(IY2,IX2))/4.0
      IF (DCP-CV) 5,217,217
  217 IF (INX(IS-1)) 214,215,214
  214 IX=IX+IDX
```

```
      IDX=-ICX
      PY=2.0
      CALL CALC
      IX=IX+ICX
      GO TO 6
  215 IY=IY+ICY
      IDY=-ICY
      PY=2.0
      CALL CALC
      IY=IY+ICY
    6 IF(AM(IY,IX-1)-CV) 306,16,16
  306 NP=NP+1
      REC(NP)=100*IX+IY
   16 IS=IS+5
      IX=IX2
      IY=IY2
      GO TO 5
   50 XT=MT
      IF(AM(IY,IX-1)-CV) 307,73,73
  307 NP=NP+1
      REC(NP)=100*IX+IY
   73 DO 74 I=1,N
      X(I)=X(I)+RC*Y(I)
   74 Y(I)=RS*Y(I)
      CALL FILTRO(IFLAG,N)
      IF(IFLAG.EQ.1) THEN
C      TYPE *,'SUB PLOTT WAS CALLED: N..',N,'  CV..',CV
      CALL PLOTT(N,CV)
      END IF
      RETURN
      END
C
C*****************************************************************************
C
      SUBROUTINE CALC
      DIMENSION REC(1900),  X(4800), Y(4800)
      COMMON/ELI/AM(256,256)
      DIMENSION IPT(3,3),INX(8),INY(8)
      COMMON /DAYHOF/ MT,NT,NI,IX,IY,IDX,IDY,ISS,IT,IV,NP,N ,JT,
     1                PY,REC,CV,    IPT,INX,INY,DL,RA,THE
      COMMON /INTFAC/ X,Y
      IT=0
      N=N+1
      IF (IDX**2 + IDY**2 -1) 20,1,20
    1 IF (IDX) 10,2,10
    2 X(N)=IX
      Z=IY
      IY2=IY+IDY
```

```
      DY=IDY
   41 Y(N)=((AM(IY,IX)-CV)/(AM(IY,IX)-AM(IY2,IX)))*DY+Z
      RETURN
   10 Y(N)=IY
      W=IX
      DX=IDX
      IX2=IX+IDX
   44 X(N)=((AM(IY,IX)-CV)/(AM(IY,IX)-AM(IY,IX2)))*DX+W
      RETURN
   20 IX2=IX+IDX
      IY2=IY+IDY
      W=IX
      Z=IY
      DX=IDX
      DY=IDY
      DCP=(AM(IY,IX)+AM(IY,IX2)+AM(IY2,IX)+AM(IY2,IX2))/4.0
      IF (PY-2.0) 24,21,24
   24 IF (DCP-CV) 21,21,25
   21 AL=AM(IY,IX)-DCP
   23 V=.5*(AL+DCP -CV)/AL
   27 X(N)=V*DX+W
      Y(N)=V*DY+Z
      PY=0.0
      RETURN
   25 IT=1
      AL=AM(IY2,IX2)-DCP
   33 V=.5*(AL+DCP-CV)/AL
   28 X(N)=-V*DX+W +  DX
      Y(N)=-V*DY+Z +  DY
      Y(N)=-V*DY+Z +  DY
      RETURN
      END
C
C**********************************************************************
C
      SUBROUTINE PLOTT(NP,CV)
      COMMON/INTFAC/X(4800),Y(4800)
      COMMON/TINA/IMAT(0:255,0:255)
      COMMON/FEFE/LC
      COMMON/FITA/NSAME
      INTEGER ICH(26)
      COMMON/FIB/ICH
      LOGICAL*1 MINUS,LABL
      COMMON/TABL/ TABC(20,6),JC
      COMMON/DITS/XMIN,YMIN,SLOPEX,SLOPEY,DITSDX,DITSDY,IDIK,LABL,MINUS
      INTEGER DX,DY,FX,FY,I
      INTEGER VECTX(4800),VECTY(4800),TOTK,K,ITYPE,THRES
      COMMON/ISABEL/THRES
```

```
      DATA (ICH(I),I=1,26)/'A','B','C','D','E','F','G','H','I','J','K',
     1  'L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'/
C      SCALE POINTS FOR PLOT ROUTINE
       X(1)=SLOPEX*(X(1)-XMIN)
       Y(1)=SLOPEY*(Y(1)-1.0)
       FX=NINT(X(1))
       FY=NINT(Y(1))
       VECTX(2)=FX
       VECTY(2)=FY
       K=2
       RFX=FLOAT(FX)
       RFY=FLOAT(FY)
C       CALL MOVEA(RFX,RFY)
       IMAT(FX,FY)=ICH(LC)
C       WRITE(6,90) FX,FY
       DO   I=2,NP
          X(I)  =SLOPEX*(X(I)-XMIN)
          Y(I) = SLOPEY * (Y(I) - 1.0)
          IX=NINT(X(I))
          IY=NINT(Y(I))
          DX=ABS(FX-IX)
          DY=ABS(FY-IY)
C CONSTRUCT THE BUFFER OF CONTOURS
          IF(.NOT.((DX.EQ.0).AND.(DY.EQ.0))) THEN
             K=K+1
             VECTX(K)=IX
             VECTY(K)=IY
          END IF
C COMPLETE THE CHARACTER MATRIX OF THE CONTOURS
          IF(((DX.EQ.0).OR.(DX.EQ.1)).AND.((DY.EQ.0).OR.(DY.EQ.1)))THEN
              CALL CONEX(FX,FY,IX,IY)
           END IF
          IF(((DX.NE.0).AND.(DX.NE.1)).AND.((DY.EQ.0).OR.(DY.EQ.1)))THEN
              CALL COMPX(FX,FY,IX,IY)
           END IF
          IF(((DX.EQ.0).OR.(DX.EQ.1)).AND.((DY.NE.0).AND.(DY.NE.1)))THEN
              CALL COMPY(FX,FY,IX,IY)
           END IF
          IF(((DX.NE.0).AND.(DX.NE.1)).AND.((DY.NE.0).AND.(DY.NE.1)))THEN
              CALL COMPXY(FX,FY,IX,IY)
           END IF
           RFX=FLOAT(IX)
           RFY=FLOAT(IY)
C           CALL DRAWA(RFX,RFY)
           FX=IX
           FY=IY
            IGAP=I-1
C           WRITE(6,50) IGAP
```

```
C50          FORMAT(1X,'END OF GAP: ',I5)
         END DO
C STORE THE CONTOURS WITH MORE THEN 10 POINTS IN A FILE FOR020.DAT
          TOTK=K-1
          IF(TOTK.GE.THRES) THEN
            NSAME=NSAME+1
C SELECT OPEN AND CLOSE CONTOURS (ITYPE = -1 & -2)
            IF((VECTX(2).EQ.VECTX(K)).AND.(VECTY(2).EQ.VECTY(K))) THEN
              ITYPE=-2
            ELSE
              ITYPE=-1
            END IF
            TYPE *,'CONTOUR NUMBER:',NSAME
            TYPE *,'CONTOUR TYPE:',ITYPE
            TYPE *,'NUMBER OF POINTS:',TOTK
            WRITE(20,124) TOTK,ITYPE
            DO I=2,K
              WRITE(20,124) VECTX(I),VECTY(I)
            END DO
          END IF
C         WRITE(6,18)
  124     FORMAT(2I5)
C90       FORMAT(11X,2(I5,5X))
C18       FORMAT(//10X,'NEXT CONTOUR')
        RETURN
        END
C
C
        SUBROUTINE CONEX(FX,FY,IX,IY)
C
C       THIS SUBROUTINE IS CALLED WHEN THE CONSECUTIVE POINTS OF THE
C       CONTOUR ARE THE SAME OR DIFFER BY ONE UNIT AT MAXIMUM
C
        COMMON/TINA/IMAT(0:255,0:255)
        COMMON/FEFE/LC
        INTEGER ICH(26)
        COMMON/FIB/ICH
        INTEGER FX,FY,IX,IY
C
        IF (FX.EQ.IX) THEN
           IF (FY.NE.IY) THEN
              IMAT(FX,IY)=ICH(LC)
C              WRITE(6,90) FX,IY
           END IF
        ELSE
           IF (FY.EQ.IY) THEN
              IMAT(IX,FY)=ICH(LC)
C             WRITE(6,90) IX,FY
```

```
            END IF
         END IF
         IF( (FX.NE.IX).AND.(FY.NE.IY)) THEN
            IMAT(IX,FY)=ICH(LC)
            IMAT(IX,IY)=ICH(LC)
C            WRITE(6,90) IX,FY
C            WRITE(6,90) IX,IY
         END IF
C90      FORMAT(11X,2(I5,5X,'CONECTED'))
         RETURN
         END
C
C
         SUBROUTINE COMPX(FX,FY,IX,IY)
C
C        THIS SUBROUTINE COMPLETES THE CONTOUR IN X DIRECTION
C
         COMMON/TINA/IMAT(0:255,0:255)
         COMMON/FEFE/LC
         INTEGER ICH(26)
         COMMON/FIB/ICH
C
         INTEGER FX,FY,IX,IY,LDW,LUP,I
C
         IF( IX.GT.FX) THEN
            LDW=FX+1
            LUP=IX
            DO 1=LDW,LUP
               IMAT(I,FY)=ICH(LC)
C              WRITE(6,90) I,FY
            END DO
            IF( IY.NE.FY) THEN
               IMAT(LUP,IY)=ICH(LC)
C              WRITE(6,90) LUP,IY
            END IF
         ELSE
            LDW=IX
            LUP=FX-1
            DO I=LUP,LDW,-1
              IMAT(I,FY)=ICH(LC)
C             WRITE(6,90) I,FY
            END DO
            IF( IY.NE.FY) THEN
              IMAT(LDW,IY)=ICH(LC)
C             WRITE(6,90) LDW,IY
            END IF
         END IF
C90      FORMAT(11X,2(I5,5X,'X'))
```

```
      RETURN
      END
C
C
      SUBROUTINE COMPY(FX,FY,IX,IY)
C
C     THIS SUBROUTINE COMPLEMENT THE CONTOUR IN Y DIRECTION
C
      COMMON/TINA/IMAT(0:255,0:255)
      COMMON/FEFE/LC
      INTEGER FX,FY,IX,IY,LDW,LUP,I,ICH(26)
      COMMON/FIB/ICH
C
      IF( IY.GT.FY) THEN
         LDW=FY+1
         LUP=IY
         DO I=LDW,LUP
           IMAT(FX,I)=ICH(LC)
C          WRITE(6,90) FX,I
         END DO
         IF (IX.NE.FX) THEN
           IMAT(IX,LUP)=ICH(LC)
C          WRITE(6,90) IX,LUP
         END IF
      ELSE
         LDW=IY
         LUP=FY-1
         DO I=LUP,LDW,-1
           IMAT(FX,I)=ICH(LC)
C          WRITE(6,90) FX,I
         END DO
         IF (IX.NE.FX) THEN
            IMAT(IX,LDW)=ICH(LC)
C           WRITE(6,90) IX,LDW
         END IF
      END IF
C90    FORMAT(11X,2(I5,5X,'Y'))
      RETURN
      END
C
C
      SUBROUTINE COMPXY(FX,FY,IX,IY)
C
C     THIS SUBROUTINE COMPLEMENTS CONTOUR IN X AND Y DIRECTIONS
C
      COMMON/TINA/IMAT(0:255,0:255)
      COMMON/FEFE/LC
      INTEGER ICH(26)
```

```
      COMMON/FIB/ICH
      INTEGER DX,DY,ADX,ADY,SDX,SDY,REST,I,ARGX,ARGY,FLAG,FX,FY,IX,IY
C
      DX=IX-FX
      DY=IY-FY
      ADX=ABS(DX)
      ADY=ABS(DY)
C     TYPE *,'FX',FX,'FY',FY
      SDX=DX/ADX
      SDY=DY/ADY
C     WRITE(6,80) SDX,SDY
C80   FORMAT(1X,'SDX.....',I5,'SDY.....',I5)
      IF (ADX.EQ.ADY) THEN
        ARGX=FX
        ARGY=FY
        FLAG=0
        DO I=1,2*ADX
          IF (FLAG.EQ.0) THEN
            ARGX=ARGX+SDX
            IMAT(ARGX,ARGY)=ICH(LC)
C           WRITE(6,90) ARGX,ARGY
            FLAG=1
          ELSE
            ARGY=ARGY+SDY
            IMAT(ARGX,ARGY)=ICH(LC)
C           WRITE(6,90) ARGX,ARGY
              FLAG=0
          END IF
        END DO
       END IF
       IF (ADX.GT.ADY) THEN
          REST=ADX-ADY
          ARGX=FX
          ARGY=FY
          FLAG=0
C         WRITE(6,66)REST,ARGX,ARGY
C66       FORMAT(1X,'ADX GT ADY ',3(I6,2X))
          DO I=1,2*ADY
            IF (FLAG.EQ.0) THEN
               ARGX=ARGX+SDX
               IMAT(ARGX,ARGY)=ICH(LC)
C              WRITE(6,90) ARGX,ARGY
               FLAG=1
            ELSE
               ARGY=ARGY+SDY
               IMAT(ARGX,ARGY)=ICH(LC)
C              WRITE(6,90) ARGX,ARGY
               FLAG=0
```

```
                ENC IF
            END DO
            DO I=1,REST
                ARGX=ARGX+SDX
                IMAT(ARGX,ARGY)=ICH(LC)
C               WRITE(6,90) ARGX,ARGY
            END DO
        END IF
        IF (ADX.LT.ADY) THEN
            REST=ADY-ADX
            ARGX=FX
            ARGY=FY
            FLAG=0
            DO I=1,2*ADX
              IF (FLAG.EQ.0) THEN
                ARGX=ARGX+SDX
                IMAT(ARGX,ARGY)=ICH(LC)
C               WRITE(6,90) ARGX,ARGY
                FLAG=1
              ELSE
                ARGY=ARGY+SDY
                IMAT(ARGX,ARGY)=ICH(LC)
C               WRITE(6,90) ARGX,ARGY
                FLAG=0
              END IF
            END DO
            DO I=1,REST
                ARGY=ARGY+SDY
                IMAT(ARGX,ARGY)=ICH(LC)
C               WRITE(6,90) ARGX,ARGY
            END DO
        END IF
C90     FORMAT(11X,2(I5,5X,'XY'))
        RETURN
        END
C
C*******************************************************************************
C SUBROUTINE FILTRO
        SUBROUTINE FILTRO(IFLAG,N)
        DIMENSION X(4800),Y(4800)
        COMMON /INFAC/X,Y
        INTEGER U,L,X1,XN,Y1,YN
        DATA L,U/0,255/
        X1=NINT(X(1))
        XN=NINT(X(N))
        Y1=NINT(Y(1))
        YN=NINT(Y(N))
        IF((X1.EQ.XN).AND.(Y1.EQ.YN)) THEN
```

```
      IFLAG=1
      RETURN
    END IF
    IF(((X1.EQ.L).AND.(XN.EQ.L)).OR.((X1.EQ.U).AND.(XN.EQ.U)))THEN
      IFLAG=1
      RETURN
    END IF
    IF(((Y1.EQ.L).AND.(YN.EQ.L)).OR.((Y1.EQ.U).AND.(YN.EQ.U)))THEN
      IFLAG=1
      RETURN
    END IF
    IF(((X1.EQ.L).AND.(XN.EQ.U)).OR.((X1.EQ.U).AND.(XN.EQ.L)))THEN
      IFLAG=1
      RETURN
    END IF
    IF(((Y1.EQ.L).AND.(YN.EQ.U)).OR.((Y1.EQ.U).AND.(YN.EQ.L)))THEN
      IFLAG=1
      RETURN
    END IF
    IF(((Y1.EQ.L).OR.(Y1.EQ.U)).AND.((XN.EQ.L).OR.(XN.EQ.U)))THEN
      IFLAG=1
      RETURN
    END IF
    IF(((X1.EQ.L).OR.(X1.EQ.U)).AND.((YN.EQ.L).OR.(YN.EQ.U)))THEN
      IFLAG=1
       RETURN
    END IF
    IFLAG=0
    RETURN
    END
```

```
C PROGRAM THE8KNT.FOR
C
C OWNER CESAR MIRANDA
C 28/OCT/83
C
C THIS IS A FORTRAN PROGRAM THAT IMPLEMENTS THE DIERK'S ALGORITHM
C FOR AUTOMATIC KNOTS SELECTION, SO THAT A TRADE-OFF BETWEEN
C CLOSENESS OF FIT AND SMOOTHNESS IS ACHIEVED. THE THREE-DIMENSIONAL
C EFFECT IS CONSIDERED BY ASSINGING A WEIGHT TO EVERY KNOT. THESE
C WEIGHTS ARE PROPORTIONAL TO THE DENSITY OF CONTOURS IN THE
C NEIGHBORHOOD OF THE RESPECTIVE KNOT.
C
C INPUT PARAMETERS
C       FILE FOR011.DAT: FILE CONTAINING THE USER CONTROLLED PARAMETERS
C       WO (REFERENCE WEIGHT) AND DISP (INCREMENTAL WEIGHT) THIS
C       LAST PARAMETER IS MENTIONED IN THE TEXT AS DELTAW.
C       FILE FOR018.DAT: FILE CONTAINING THE CHARACTER MATRIX WHERE
C       THE DENSITY MEASUREMENTS ARE PERFORMED.
C       FILE FOR020.DAT: FILE CONTAINING THE CONTOURS REPRESENTED BY
C       THEIR (X,Y) PAIRS. THE CONTOURS ARE CONCATENED IN A TWO COLUMNS
C       RECORD. THE SEPARATION BETWEEN CONTOURS IS DONE BY A HEADER
C       THAT GIVES THE NUMBER OF POINTS IN THE FOLLOWING CONTOUR AND
C       THE TYPE OF THE CONTOUR (-1 OPENNED OR -2 CLOSE).
C
C
C OUTPUT PARAMETERS
C       FILE FOR022.DAT: FILE CONTAINING THE SELECTED KNOTS GENERATED
C       BY THE MODIFIED DIERK'S ALGORITHM. THE STRUCTURE OF THIS FILE
C       IS THE SAME AS THE INPUT FILE FOR020.DAT.
C
C VARIABLES DECLARATION
      REAL X(4800),Y(4800),T(4800),Z(4800),W(4800),
     1  CX(4800),CY(4800),FP,ZA,ZE,SPX,SPY,ARG,S
      INTEGER I,J,K,IOPT,IPAR,IER,N,M,IX,IY,ITYPE,NK1,L,NEND,
     1  TOTPT,TOTKNT,IERROR,IR,IW,CONT
      INTEGER IM(256,256),IC,WINDOW
      REAL WM,DISP,WO
      COMMON/CIDA/IM
C ASSIGN THE INPUT OUTPUT UNITS AN FIXED PARAMETERS
      DATA IR,IW/20,22/
      DATA IC,WINDOW,IBL/18,5,' '/
C INPUT THE WEIGHT PARAMETERS (REFERENCE AND INCREMENTAL)
      READ(11,50) WO,DISP
50    FORMAT(2F5.2)
      TYPE *,'WEIGTHING PARAMETER WO...',WO,'  DISP',DISP
C
C INPUT THE CHARACTER MATRIX
      DO I=1,256
```

```
            READ(IC,130)(IM(I,J),J=1,128)
            READ(IC,130)(IM(I,J),J=129,256)
         END DO
         TYPE *,'CHAR MATRIX WAS INPUT'
C SET PARAMETERS
C CUBIC SPLINE
         K=3
C AUTOMATIC KNOTS PARAMETER GENERATION
         IPAR=0
C INITIALIZATION
         IOPT=0
         TOTPT=0
         TOTKNT=0
         CONT=0
C
C GIVE THE USER THE OPTION TO JUMP TO THE DESIRED CONTOUR
         TYPE *,'ENTER THE DESIRED CONTOUR TO START WITH'
C        ACCEPT *,JUMP
         JUMP=1
         DO WHILE(JUMP.GT.1)
            READ(20,124) IX,IY
            M=IX
            DO I=1,M
               READ(20,124) IX,IY
            END DO
            JUMP=JUMP-1
         END DO
C INPUT THE HEADER OF FIRST CONTOUR
         READ(20,124) IX,IY
C INPUT CONTOUR TILL SENTINEL IS FOUND
         DO WHILE(IX.NE.-3)
           M=IX
           TOTPT=TOTPT+M
           ITYPE=IY
           CONT=CONT+1
           TYPE *,'  '
           TYPE *,'CONTOUR NUMBER',CONT,'  TYPE',ITYPE
           TYPE *,'NUMBER OF POINTS',M
           IF(M.GT.4800) THEN
           IERROR=-30
           TYPE *,'ERROR',IERROR
           STOP
           END IF
           DO I=1,M
             READ(20,124) IX,IY
             X(I)=FLOAT(IX)
             Y(I)=FLOAT(IY)
C SET THE WEIGTH
```

```
            IF(IM(IX+1,IY+1).EQ.IBL) THEN
               TYPE *,'TRAP..NC CORRESPONDENCE'
               STOP
            END IF
            IF((IX.GT.5).AND.(IX.LT.252).AND.(IY.GT.5).AND.(IY.LT.252))
     1      THEN
               CALL TRIDI(IX,IY,WINDOW,W0,DISP,WM)
               W(I)=WM
            ELSE
               W(I)= W0
            END IF
C           TYPE *,'WM...',WM
         END DO
         TYPE *,IX,IY
C CALL DIERKS ROUTINES SELECTED BY ITYPE
C        TYPE *,'INPUT THE SMOOTHING FACTOR S'
C        ACCEPT *,S
         MFA=M/100+1
         S=SQRT(FLOAT(M))*FLOAT(MFA)
         TYPE *,'VALUE OF S',S
         IF (ITYPE.EQ.-1) THEN
           CALL PARAM(X,Y,Z,W,M,LB,LE,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
         ELSE
           CALL PERCLO(X,Y,Z,W,M,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
         END IF
C OUTPUT THE HEADER OF CONTOUR OF SELECTED KNOTS
         N6=N-6
         TOTKNT=TOTKNT+N6
         WRITE(22,124)N6,ITYPE
         TYPE *,'IER....',IER
C EVALUATE AND STORE THE KNOTS X & Y OF THE CURRENT CONTOUR IN A 215
C FORMAT
         NK1=N-K-1
         L=K+1
         NEND=N-K
         DO I=L,NEND
           ARG=T(I)
C SEARCH FOR KNOT INTERVAL T(L)<=ARG<T(L+1)
300        IF((ARG.LT.T(L+1)).OR.(L.EQ.NK1)) GO TO 400
           L=L+1
           GO TO 300
400        CONTINUE
           SPX=DERIV(T,N,CX,NK1,0,ARG,L)
           SPY=DERIV(T,N,CY,NK1,0,ARG,L)
           IX=NINT(SPX)
           IY=NINT(SPY)
           WRITE(22,124) IX,IY
         END DO
```

```
        TYPE *,'NUMBER OF KNOTS',N
        READ(20,124) IX,IY
      END DO
      TYPE *,'END OF DATA, SENTINEL WAS FOUND'
      IX=-3
      IY=-3
      WRITE(22,124) IX,IY
      TYPE *,'TOTAL NUMBER OF INPUT POINTS',TOTPT
      TYPE *,'TOTAL NUMBER OF OUTPUT KNOTS',TOTKNT
      STOP
124   FORMAT(2I5)
130   FORMAT(128A1)
      END
```

```
      SUBROUTINE TRIDI(IX,IY,WINDOW,WO,DISP,WM)
C
C 28/OCT/83
C OWNER CESAR MIRANDA
C
C THIS SUBROTINE SET THE WEIGTHS FOR DIERKS ALGORITHM IN ORDER TO
C ADD THE TRI DIMENSIONAL EFFECT IN THE KNOTS SELECTION
C
C INPUT PARAMETERS
C  IX,IY : COORDINATES OF THE GIVEN POINTS RANGE _0..255'
C  WINDOW: MAXIMUM DISPLACEMENT FROM THE GIVEN POINT (IX,IY)
C    TO SEARCH FOR DENSITY OF CONTOURS
C  DISP : INCREMENT IN THE WEIGHT FOR EACH CONTOUR FOUND
C    IN THE CURRENT SEARCH AREA
C
C VARIABLE DECLARATION
      INTEGER IM(256,256),IBL,ICHA,I,WINDOW
      REAL WE,WN,WW,WS,WO,DISP,WM
      COMMON/CIDA/IM
C
      DATA IBL/' '/
C
C      TYPE *,'SET INITIAL WEIGHT'
      WE=WO
      WN=WO
      WW=WO
      WS=WO
C SHIFTING BY 1 THE GIVEN COORDINATES TO RANGE _1..256'
      IX1=IX+1
      IY1=IY+1
      ICHA=IM(IX1,IY1)
      DO I=1,WINDOW
         IE=IM(IX1,IY1+I)
         IN=IM(IX1-I,IY1)
         IW=IM(IX1,IY1-I)
         IS=IM(IX1+I,IY1)
         IF((IE.NE.ICHA).AND.(IE.NE.IBL)) THEN
           WE=WE+DISP
C          TYPE *,'WE',WE
         END IF
         IF((IN.NE.ICHA).AND.(IN.NE.IBL)) THEN
            WN=WN+DISP
C           TYPE *,'WN',WN
         END IF
         IF((IW.NE.ICHA).AND.(IW.NE.IBL)) THEN
            WW=WW+DISP
C           TYPE *,'WW',WW
         END IF
```

```
            IF((IS.NE.ICHA).AND.(IS.NE.IBL)) THEN
               WS=WS+DISP
C              TYPE *,'WS',WS
            END IF
         END DO
         WM=AMAX1(WE,WN,WW,WS)
C        TYPE *,'FROM SUBROUTINE WM',WM
         RETURN
         END
```

```
      SUBROUTINE PARAM(X,Y,Z,W,M,ZB,ZE,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
C   GIVEN THE SET OF DATA POINTS (X(I),Y(I)) WITH CORRESPONDING Z-
C   VALUES Z(I),I=1,2,...M AND GIVEN ALSO THE SET OF POSITIVE
C   NUMBERS W(I),I=1,2,...M, SUBROUTINE PARAM FINDS A SMOOTH APPROXIMAT-
C   ING CURVE WITH PARAMETER REPRESENTATION X = SX(Z), Y = SY(Z).
C   SX(Z) AND SY(Z) ARE TWO SPLINE FUNCTIONS OF DEGREE K WITH THE NUMBER
C   AND THE POSITION OF THE KNOTS T(J),J=1,2,...N AUTOMATICALLY
C   CHOSEN BY THE ROUTINE. THE SMOOTHNESS OF SX(Z) AND SY(Z) IS
C   ACHIEVED BY MINIMALIZING THE SUM(DX(R)**2+DY(R)**2) WHERE DX(R)
C   AND DY(R) STAND FOR THE DISCONTINUITYJUMP OF THE KTH DERIVATIVE
C   OF SX(Z) AND SY(Z) AT THE KNOT T(R), R=K+2,...N-K-1.
C   THE AMOUNT OF SMOOTHNESS IS DETERMINED BY THE CONDITION THAT F(P) =
C   SUM(W(I)*((X(I)-SX(Z(I)))**2 +(Y(I)-SY(Z(I)))**2)) BE <= S, WITH
C   S A GIVEN NON-NEGATIVE CONSTANT.
C   THE SPLINE FUNCTIONS SX(Z) AND SY(Z) ARE GIVEN IN THEIR B-SPLINE
C   REPRESENTATION (B-SPLINE COEFFICIENTS CX(J),RESP. CY(J),J=1,...N-K-1)
C   AND CAN BE EVALUATED BY MEANS OF FUNCTION DERIV.
C   CALLING SEQUENCE:
C      CALL PARAM(X,Y,Z,W,M,ZB,ZE,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
C
C   INPUT PARAMETERS:
C     X     : ARRAY,LENGTH M, CONTAINING THE ABSCISSAE OF THE DATA POINTS
C     Y     : ARRAY,LENGTH M, CONTAINING THE ORDINATES OF THE DATA POINTS
C     W     : ARRAY,MINIMUM LENGTH M,CONTAINING THE WEIGHTS W(I).
C     M     : INTEGER VALUE,CONTAINING THE NUMBER OF DATA POINTS.
C     K     : INTEGER VALUE,CONTAINING THE DEGREE OF SX(Z) AND SY(Z).
C     S     : REAL VALUE,CONTAINING THE SMOOTHING FACTOR.
C     IOPT  : INTEGER VALUE WHICH TAKES THE VALUE 0 OR 1.
C       IOPT=0: THE ROUTINE WILL RESTART ALL COMPUTATIONS.
C       IOPT=1: THE ROUTINE WILL START WITH THE KNOTS FOUND AT THE
C               LAST CALL OF THE ROUTINE. IF IOPT=1 THE OUTPUT
C               PARAMETERS T AND N ARE INPUT PARAMETERS AS WELL.
C               IF IOPT=1 THE USER MUST PROVIDE WITH A COMMON BLOCK
C                  COMMON/OPT1/NRDATA(NEST),FPO,FPOLD,NPLUS
C     IPAR  : INTEGER FLAG.
C       IPAR = 0: FOR EACH DATA POINT (X(I),Y(I)) THE PROGRAM AUTOMATICALLY
C                 CHOOSES A CORRESPONDING VALUE OF THE PARAMETER Z, I.E.
C                 Z(1)=0;Z(I)=Z(I-1)+SQRT((X(I)-X(I-1))**2+(Y(I)-Y(I-1))**2)
C                 THE BOUNDARIES FOR THE PARAMETER Z ARE CHOSEN AS FOLLOWS
C                   ZB = Z(1) ; ZE = Z(M).
C       IPAR = 1: THE USER HIMSELF PROVIDES WITH THE VALUES OF THE
C                 PARAMETER Z AND WITH THE BOUNDARIES ZB AND ZE.
C     Z     : ARRAY,LENGTH M, CONTAINING THE VALUES OF THE PARAMETER Z
C             (IPAR = 1)
C     ZB,ZE: REAL VALUES, CONTAINING THE BOUNDARIES OF THE PARAMETER Z
C             (IPAR = 1).
C
C   OUTPUT PARAMETERS:
```

```
C     T     : ARRAY,LENGTH NEST (SEE DATA INITIALIZATION STATEMENT),
C             WHICH CONTAINS THE POSITION OF THE KNOTS,I.E. THE POSITION
C             OF THE INTERIOR KNOTS T(K+2),...T(N-K-1), AS WELL AS THE
C             POSITION OF THE KNOTS T(1)=T(2)=...=T(K+1)=ZB AND ZE =
C             T(N-K)=...=T(N) WHICH ARE NEEDED FOR THE B-SPLINE REPRESENT.
C     CX,CY: ARRAYS,LENGTH NEST, CONTAINING THE B-SPLINE COEFFICIENTS
C             OF SX(Z), RESP. SY(Z).
C     N     : INTEGER VALUE,CONTAINING THE TOTAL NUMBER OF KNOTS.
C     FP    : REAL VALUE, WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2)
C             + SUM(WI*(YI-SY(ZI))**2), I=1,2,...M.
C     IER   : ERROR CODE
C       IER=0: NORMAL RETURN.
C       IER=-1:NORMAL RETURN,SX(Z) AND SY(Z) ARE INTERPOLATING SPLINES
C       IER=-2:NORMAL RETURN, SX(Z) AND SY(Z) ARE POLYNOMIALS OF DEGREE K
C       IER>0 :ABNORMAL TERMINATION.
C         IER=1:THE REQUIRED STORAGE SPACE EXCEEDS THE AVAILABLE
C               STORAGE SPACE,SPECIFIED BY THE PARAMETER NEST.
C               PROBABLY CAUSES:NEST OR S TOO SMALL.
C         IER=2:A THEORETICALLY IMPOSSIBLE RESULT WAS FOUND DURING
C               THE ITERATION PROCESS.
C               PROBABLY CAUSES:TOL TOO SMALL
C         IER=3:THE MAXIMAL NUMBER OF ITERATIONS MAXIT HAS BEEN
C               REACHED.
C               PROBABLY CAUSES:MAXIT OR TOL TOO SMALL.
C         IER=10:SOME OF THE INPUT DATA ARE INVALID(SEE RESTRICTIONS).
C
C   RESTRICTIONS:
C     1) M > K > 0
C     2) ZB <= Z(R) < Z(R+1) <= ZE, R=1,2,...M-1.  (IPAR = 1)
C     3) W(R) > 0, R=1,2,...M.
C     4) S >= 0.
C     5) NEST >= 2*K+2.
C
C   OTHER SUBROUTINES REQUIRED:
C     BSPLIN,CCSSIN,ROTATE,BACK,NKNOT,DISCO AND RATION.
      DIMENSION X(M),Y(M),W(M),Z(M),T(1000),CX(1000),CY(1000),
     < FPINT(1000),RX(1000),RY(1000),DIAG(1000),DPRIME(1000),
     < G(1000,6),B(1000,7),Q(2000,6),H(7),NRDATA(1000),A(1000,5)
C   COMMON/OPT1/NRDATA(NEST),FP0,FPOLD,NPLUS
C     NRDATA: INTEGER ARRAY,LENGTH NEST,WHICH GIVES THE NUMBER OF
C             DATA POINTS INSIDE EACH KNOT INTERVAL.
C     FP0   : REAL VALUE, WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2)+
C             SUM(WI*(YI-SY(ZI))**2) WITH SX(Z) AND SY(Z) LEAST-SQUARES
C             POLYNOMIALS OF DEGREE K.
C     FPOLD : REAL VALUE,WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2)+
C             SUM(WI*(YI-SY(ZI))**2) WITH SX(Z) AND SY(Z) LEAST-SQUARES
C             SPLINE FUNCTIONS CORRESPONDING TO THE LAST FOUND SET OF
C             KNOTS BUT ONE.
```

```
C     NPLUS : INTEGER VALUE,GIVING THE NUMBER OF KNOTS OF THE LAST
C             SET MINUS THE NUMBER OF THE LAST SET BUT ONE.
      COMMON/CPT1/NRDATA,FPO,FPOLD,NPLUS
C   DATA INITIALIZATION STATEMENT TO SPECIFY
C     TOL   : THE REQUESTED RELATIVE ACCURACY FOR THE ROOT OF F(P) = S.
C     MAXIT: THE MAXIMAL NUMBER OF ITERATIONS ALLOWED.
C     NEST : AN OVER-ESTIMATE OF THE NUMBER OF KNOTS N. THIS PARAMETER
C            MUST BE SET BY THE USER TO INDICATE THE STORAGE SPACE
C            AVAILABLE TO THE SUBROUTINE. THE DIMENSION SPECIFICATIONS
C            OF THE ARRAYS  T,CX,CY,NRDATA,FPINT,RX,RY,DIAG,DPRIME(N),
C            A(N,K),G(N,K+1),B(N,K+2),Q(M,K+1) AND H(K+2) DEPEND
C            ON N,M AND K. SINCE N IS UNKNOWN AT THE TIME THE
C            USER SETS UP THE DIMENSION INFORMATION AN OVER-ESTIMATE
C            OF THESE ARRAYS WILL GENERALLY BE MADE. THE FOLLOWING
C            REMARKS ARE INTENDED TO HELP THE USER
C               1) 2*K+2 <= N <= M+K+1
C               2) THE SMALLER THE VALUE OF S, THE GREATER N WILL BE.
C               3) NORMALLY N = M/2 IS AN OVER-ESTIMATE.
      DATA TOL/0.001/,MAXIT/20/,NEST/1000/
C   BEFORE STARTING COMPUTATIONS A DATA CHECK IS MADE. IF THE INPUT
C   DATA ARE INVALID CONTROLE IS IMMEDIATELY REPASSED TO THE DRIVER
C   PROGRAM (IER=10).
      IER = 0
      K1  = K+1
      K2  = K1+1
      NMIN = 2*K1
      IF(K.LE.0) IER = 10
      IF(M.LT.K1 .OR. NEST.LT.NMIN) IER = 10
      IF(S.LT.0.) IER = 10
      IF(IER.NE.0) GO TO 440
C   CHECK WHETHER THE Z-VALUES ARE PROVIDED WITH BY THE USER.
      IF(IPAR.NE.0) GO TO 6
C   FIND FOR EACH DATA POINT A CORRESPONDING VALUE OF THE PARAMETER Z
C   AND FIX THE BOUNDARIES ZB AND ZE.
      Z(1) = 0.
      DO 4 I=2,M
        Z(I) = Z(I-1)+SQRT((X(I)-X(I-1))**2+(Y(I)-Y(I-1))**2)
    4 CONTINUE
      ZB = Z(1)
      ZE = Z(M)
    6 IF(ZB.GT.Z(1) .OR. ZE.LT.Z(M) .OR. W(1).LE.0.) IER = 10
      DO 10 I=2,M
        IF(Z(I-1).GE.Z(I) .OR. W(I).LE.0.) IER = 10
   10 CONTINUE
      IF(IER.NE.0) GO TO 440
C   CALCULATION OF ACC, THE ABSOLUTE TOLERANCE FOR THE ROOT OF F(P)=S.
      ACC = TOL*S
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
C   PART 1: DETERMINATION OF THE NUMBER OF KNOTS AND THEIR POSITION     C
C   ************************************************************     C
C   GIVEN A SET OF KNOTS WE COMPUTE THE LEAST-SQUARES SPLINES SXINF(Z)  C
C   AND SYINF(Z).IF THE SUM F(P=INF)<=S WE ACCEPT THE CHOICE OF KNOTS.  C
C   OTHERWISE WE HAVE TO INCREASE THEIR NUMBER.                         C
C   THE INITIAL CHOICE OF KNOTS DEPENDS ON THE VALUE OF S AND IOPT.     C
C     IF S=0 WE HAVE SPLINE INTERPOLATION; IN THAT CASE THE NUMBER OF   C
C     KNOTS EQUALS NMAX = M+K+1.                                        C
C     IF S > 0 AND                                                      C
C       IOPT=0 WE FIRST COMPUTE THE LEAST-SQUARES POLYNOMIALS OF        C
C       DEGREE K; N = NMIN = 2*K+2                                      C
C       IOPT=1 WE START WITH THE SET OF KNOTS FOUND AT THE LAST         C
C       CALL OF THE ROUTINE, EXCEPT FOR THE CASE THAT S > FP0; THEN     C
C       WE COMPUTE DIRECTLY THE LEAST-SQUARES POLYNOMIALS OF DEGREE K.  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   DETERMINE NMAX, THE NUMBER OF KNOTS FOR SPLINE INTERPOLATION.
      NMAX = M+K1
      IF(S.GT.0.) GO TO 45
C   IF S=0, SX(Z) AND SY(Z) ARE INTERPOLATING SPLINES.
      N = NMAX
C   TEST WHETHER THE REQUIRED STORAGE SPACE EXCEEDS THE AVAILABLE ONE.
      IF(N.GT.NEST) GO TO 420
C   FIND THE POSITION OF THE INTERIOR KNOTS IN CASE OF INTERPOLATION.
      MK1 = M-K1
      IF(MK1.EQ.0) GO TO 60
      K3 = K/2
      I = K2
      J = K3+2
      IF(K3*2.EQ.K) GO TO 30
      DO 20 L=1,MK1
        T(I) = Z(J)
        I = I+1
        J = J+1
   20 CONTINUE
      GO TO 60
   30 DO 40 L=1,MK1
        T(I) = (Z(J)-Z(J-1))*0.5
        I = I+1
        J = J+1
   40 CONTINUE
      GO TO 60
C   IF S>0 OUR INITIAL CHOICE OF KNOTS DEPENDS ON THE VALUE OF IOPT.
C   IF IOPT=0 OR IOPT=1 AND S>=FP0, WE START COMPUTING THE LEAST-SQUARES
C   POLYNOMIALS OF DEGREE K WHICH ARE SPLINES WITHOUT INTERIOR KNOTS.
C   IF IOPT=1 AND FP0>S WE START COMPUTING THE LEAST-SQUARES SPLINES
C   ACCORDING TO THE SET OF KNOTS FOUND AT THE LAST CALL OF THE ROUTINE.
   45 IF(IOPT.LE.0) GO TO 50
      IF(FP0.GT.S) GO TO 60
```

```
   50  N = NMIN
       NRDATA(1) = M-2
C   MAIN LOOP FOR THE DIFFERENT SETS OF KNOTS. M IS A SAVE UPPER BOUND
C   FOR THE NUMBER OF TRIALS.
   60  DO 200 ITER = 1,M
         IF(N.EQ.NMIN) IER = -2
C   FIND NRINT, THE NUMBER OF KNOT INTERVALS.
         NRINT = N-NMIN+1
C   FIND THE POSITION OF THE ADDITIONAL KNOTS WHICH ARE NEEDED FOR
C   THE B-SPLINE REPRESENTATION OF SX(Z) AND SY(Z).
         NK1 = N-K1
         I = N
         DO 70 J=1,K1
           T(J) = ZB
           T(I) = ZE
           I = I-1
   70    CONTINUE
C   COMPUTE THE B-SPLINE COEFFICIENTS OF THE LEAST-SQUARES SPLINES SXINF(Z)
C   AND SYINF(Z). THE OBSERVATION MATRIX A IS BUILT UP ROW BY ROW AND
C   REDUCED TO UPPER TRIANGULAR FORM BY GIVENS TRANSFORMATIONS
C   WITHOUT SQUARE ROOTS. AT THE SAME TIME FP=F(P=INF) IS COMPUTED
         FP = 0.
C   INITIALIZE THE OBSERVATION MATRIX A.
         DO 80 I=1,NK1
           DIAG(I) = 0.
           RX(I) = 0.
           RY(I) = 0.
           DO 80 J=1,K
             A(I,J) = 0.
   80    CONTINUE
         L = K1
         DO 130 IT=1,M
C   FETCH THE CURRENT DATA POINT X(IT),Y(IT),Z(IT).
           XI = X(IT)
           YI = Y(IT)
           ZI = Z(IT)
           WI = W(IT)
C   SEARCH FOR KNOT INTERVAL T(L) <= ZI <= T(L+1).
           IF(ZI.GE.T(L+1) .AND. L.NE.NK1) L = L+1
C   EVALUATE THE (K+1) NON-ZERO B-SPLINES AT ZI AND STORE THEM IN Q.
           CALL BSPLIN(T,N,K,ZI,L,H)
           DO 90 I=1,K1
             IF(H(I).LT.0.1E-07) H(I) = 0.
             Q(IT,I) = H(I)
   90      CONTINUE
C   ROTATE THE NEW ROW OF THE OBSERVATION MATRIX INTO TRIANGLE BY
C   GIVENS TRANSFORMATIONS WITHOUT SQUARE ROOTS.
           J = L-K1
```

```
            DO 110 I=1,K1
              IF(WI.EQ.0.) GO TO 130
              J = J+1
              PIV = H(I)
              IF(PIV.EQ.0.) GO TO 110
C  CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
              CALL COSSIN(PIV,WI,DIAG(J),COS,SIN)
C  TRANSFORMATIONS TO RIGHT HAND SIDES.
              CALL ROTATE(PIV,COS,SIN,XI,RX(J))
              CALL ROTATE(PIV,COS,SIN,YI,RY(J))
              IF(I.EQ.K1) GO TO 120
              I2 = 0
              I3 = I+1
              DO 100 I1 = I3,K1
                I2 = I2+1
C  TRANSFORMATIONS TO LEFT HAND SIDE.
                CALL ROTATE(PIV,COS,SIN,H(I1),A(J,I2))
  100         CONTINUE
  110       CONTINUE
C  ADD CONTRIBUTION OF THIS ROW TO THE SUM OF SQUARES OF RESIDUAL
C  RIGHT HAND SIDES.
  120       FP = FP+WI*(XI**2+YI**2)
  130     CONTINUE
          IF(IER.EQ.-2) FP0 = FP
C  BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFFICIENTS.
          CALL BACK(A,RX,NK1,K,CX)
          CALL BACK(A,RY,NK1,K,CY)
C  TEST WHETHER THE APPROXIMATION X=SXINF(Z),Y=SYINF(Z) IS AN
C  ACCEPTABLE SOLUTION.
          FPMS = FP-S
          IF(ABS(FPMS).LT.ACC) GO TO 440
C  IF F(P=INF) < S ACCEPT THE CHOICE OF KNOTS.
          IF(FPMS.LT.0.) GO TO 250
C  IF N=NMAX,SXINF(Z) AND SYINF(Z) ARE INTERPOLATING SPLINES.
          IF(N.EQ.NMAX) GO TO 430
C  INCREASE THE NUMBER OF KNOTS.
C  IF N=NEST WE CANNOT INCREASE THE NUMBER OF KNOTS BECAUSE OF
C  THE STORAGE CAPACITY LIMITATION.
          IF(N.EQ.NEST) GO TO 420
C  DETERMINE THE NUMBER OF KNOTS NPLUS WE ARE GOING TO ADD.
          IF(IER.EQ.0) GO TO 140
          NPLUS = 1
          IER = 0
          GO TO 150
  140     NPL1 = NPLUS*2
          IF(FPOLD-FP.GT.ACC) NPL1 = FLOAT(NPLUS)*FPMS/(FPOLD-FP)
          NPLUS = MINO(NPLUS*2,MAXO(NPL1,NPLUS/2,1))
  150     FPOLD = FP
```

```
C  COMPUTE THE SUM(W1*((X1-SXINF(ZI))**2+(Y1-SYINF(ZI))**2)) FOR
C  EACH KNOT INTERVAL T(J+K) <= ZI <= T(J+K+1) AND STORE IT IN
C  FPINT(J),J=1,2,...NRINT.
      FPART = .0.
      I = 1
      L = K2
      NEW = 0
      DO 180 IT=1,M
        IF(Z(IT).LT.T(L) .OR. L.GT.NK1) GO TO 160
        NEW = 1
        L = L+1
160     TERM1 = 0.
        TERM2 = 0.
        LO = L-K2
        DO 170 J=1,K1
          LC = LO+1
          TERM1 = TERM1+CX(LO)*Q(IT,J)
          TERM2 = TERM2+CY(LO)*Q(IT,J)
170     CONTINUE
        TERM = W(IT)*((TERM1-X(IT))**2+(TERM2-Y(IT))**2)
        FPART = FPART+TERM
        IF(NEW.EQ.0) GO TO 180
        STORE = TERM*0.5
        FPINT(I) = FPART-STORE
        I = I+1
        FPART = STORE
        NEW = 0
180   CONTINUE
      FPINT(NRINT) = FPART
      DO 190 L=1,NPLUS
C  ADD A NEW KNOT.
        CALL NKNOT(Z,M,T,N,FPINT,NRDATA,NKINT)
C  TEST WHETHER WE CANNOT FURTHER INCREASE THE NUMBER OF KNOTS.
        IF(N.EQ.NMAX .OR. N.EQ.NEST) GO TO 200
190   CONTINUE
C  RESTART THE COMPUTATIONS WITH THE NEW SET OF KNOTS.
200   CONTINUE
C  TEST WHETHER THE APPROXIMATION X=SX(Z),Y=SY(Z) WITH SX(Z) AND SY(Z)
C  THE LEAST-SQUARES KTH DEGREE POLYNOMIALS, IS A SOLUTION OF OUR PROBLEM
250   IF(IER.EQ.-2) GO TO 440
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  PART 2: DETERMINATION OF THE SMOOTHING SPLINES SXP(Z) AND SYP(Z).      C
C  **********************************************************************C
C  WE HAVE DETERMINED THE NUMBER OF KNOTS AND THEIR POSITION.             C
C  WE NOW COMPUTE THE B-SPLINE COEFFICIENTS OF THE SMOOTHING SPLINES       C
C  SXP(Z) AND SYP(Z). THE OBSERVATION MATRIX A IS EXTENDED BY THE ROWS    C
C  OF MATRIX B EXPRESSING THAT THE KTH DERIVATIVE DISCONTINUITIES OF       C
C  SXP(Z) AND SYP(Z) AT THE INTERIOR KNOTS T(K+2),...T(N-K-1) MUST BE     C
```

```
C   ZERO. THE CORRESPONDING WEIGHTS OF THESE ADDITIONAL ROWS ARE SET     C
C   TO 1/SQRT(P). ITERATIVELY WE THEN HAVE TO DETERMINE THE VALUE OF P   C
C   SUCH THAT F(P)=SUM(WI*((XI-SXP(ZI))**2+(YI-SYP(ZI))**2) BE = S. WE   C
C   ALREADY KNOW THAT THE LEAST-SQUARES POLYNOMIALS CORRESPOND TO P=0,    C
C   AND THAT THE LEAST-SQUARES SPLINES CORRESPOND TO P=INFINITY. THE      C
C   ITERATION PROCESS WHICH IS PROPOSED HERE, MAKES USE OF RATIONAL       C
C   INTERPOLATION. SINCE F(P) IS A CONVEX AND STRICTLY DECREASING         C
C   FUNCTION OF P, IT CAN BE APPROXIMATED BY A RATIONAL FUNCTION          C
C   R(P) = (U*P+V)/(P+W). THREE VALUES OF P(P1,P2,P3) WITH CORRESPOND-    C
C   ING VALUES OF F(P) (F1=F(P1)-S,F2=F(P2)-S,F3=F(P3)-S) ARE USED        C
C   TO CALCULATE THE NEW VALUE OF P SUCH THAT R(P)=S. CONVERGENCE IS      C
C   GUARANTEED BY TAKING F1>0 AND F3<0.                                   C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   EVALUATE THE DISCONTINUITY JUMP OF THE KTH DERIVATIVE OF THE
C   B-SPLINES AT THE KNOTS T(L),L=K+2,...N-K-1 AND STORE IN B.
      CALL DISCO(T,N,K2,B)
C   INITIAL VALUE FOR P.
      P1 = 0.
      F1 = FPC-S
      P3 = -1.
      F3 = FPMS
      P = -F1/F3
      ICHECK = 0
      N8 = N-NMIN
C   ITERATION PROCESS TO FIND THE ROOT OF F(P) = S.
      DO 350 ITER=1,MAXIT
C   THE ROWS OF MATRIX B WITH WEIGHT 1/SQRT(P) ARE ROTATED INTO
C   THE TRIANGULARISED OBSERVATION MATRIX A WHICH IS STORED IN G.
        PINV = 1.0/P
        DO 260 I=1,NK1
          DPRIME(I) = DIAG(I)
          CX(I) = RX(I)
          CY(I) = RY(I)
          G(I,K1) = 0.
          DO 260 J=1,K
            G(I,J) = A(I,J)
 260    CONTINUE
        DO 300 IT=1,N8
C   THE ROW OF MATRIX B IS ROTATED INTO TRIANGLE BY GIVENS TRANSFORMATIONS.
          DO 270 I=1,K2
            H(I) = B(IT,I)
 270      CONTINUE
          XI = 0.
          YI = 0.
          WI = PINV
          DO 290 J=IT,NK1
            IF(WI.EQ.0.) GO TO 300
            PIV = H(1)
```

```
C  CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
            CALL COSSIN(PIV,WI,DPRIME(J),COS,SIN)
C  TRANSFORMATIONS TO RIGHT HAND SIDES.
            CALL ROTATE(PIV,COS,SIN,XI,CX(J))
            CALL ROTATE(PIV,COS,SIN,YI,CY(J))
            IF(J.EQ.NK1) GO TO 300
            I2 = K1
            IF(J.GT.N8) I2 = NK1-J
            DO 280 I=1,I2
C  TRANSFORMATIONS TO LEFT HAND SIDE.
              CALL ROTATE(PIV,COS,SIN,H(I+1),G(J,1))
              H(I) = H(I+1)
 280        CONTINUE
            H(I2+1) = 0.
 290      CONTINUE
 300    CONTINUE
C  BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFFICIENTS.
        CALL BACK(G,CX,NK1,K,CX)
        CALL BACK(G,CY,NK1,K,CY)
C  COMPUTATION OF F(P).
        FP = 0.
        L = K2
        DO 330 IT=1,M
          IF(Z(IT).LT.T(L) .OR. L.GT.NK1) GO TO 310
          L = L+1
 310      LO = L-K2
          TERM1 = 0.
          TERM2 = 0.
          DO 320 J=1,K1
            LO = LO+1
            TERM1 = TERM1+CX(LO)*Q(IT,J)
            TERM2 = TERM2+CY(LO)*Q(IT,J)
 320      CONTINUE
          FP = FP+W(IT)*((TERM1-X(IT))**2+(TERM2-Y(IT))**2)
 330    CONTINUE
C  TEST WHETHER THE APPROXIMATION X=SXP(Z),Y=SYP(Z) IS AN ACCEPTABLE
C  SOLUTION.
        FPMS = FP-S
        IF(ABS(FPMS).LT.ACC) GO TO 440
C  TEST WHETHER THE MAXIMAL NUMBER OF ITERATIONS IS REACHED.
        IF(ITER.EQ.MAXIT) GO TO 400
C  CARRY OUT ONE MORE STEP OF THE ITERATION PROCESS.
        P2 = P
        F2 = FPMS
        IF(ICHECK.NE.0) GO TO 340
        IF((F2-F3).GT.ACC) GO TO 335
C  OUR INITIAL CHOICE OF P IS TOO LARGE.
        P = P*0.1E-02
```

```
            P3 = F2
            F3 = F2
            GO TO 350
  335       IF((F1-F2).GT.ACC) GO TO 340
C   OUR INITIAL CHOICE OF P IS TOO SMALL
            TYPE *,'VALUE OF P',P
            P = P*0.1E+04
            P1 = F2
            F1 = F2
            GO TO 350
C   TEST WHETHER THE ITERATION PROCESS PROCEEDS AS THEORETICALLY
C   EXPECTED.
  340       IF(F2.GE.F1 .OR. F2.LE.F3) GO TO 410
            ICHECK = 1
C   FIND THE NEW VALUE FOR P.
            P = RATION(P1,F1,P2,F2,P3,F3)
  350   CONTINUE
C   ERROR CODES AND MESSAGES.
  400   IER = 3
        GO TO 440
  410   IER = 2
        GO TO 440
  420   IER = 1
        GO TO 440
  430   IER = -1
  440   RETURN
        END
        SUBROUTINE BSPLIN(T,N,K,X,L,H)
C   SUBROUTINE BSPLIN EVALUATES THE (K+1) NON-ZERO B-SPLINES OF
C   DEGREE K AT T(L) <= X < T(L+1) USING THE STABLE RECURRENCE
C   RELATION OF DE BOOR AND COX.
C   THE DIMENSION SPECIFICATIONS OF THE FOLLOWING ARRAYS MUST BE
C   AT LEAST H(K+1),HH(K).
        DIMENSION T(N),H(6),HH(5)
        H(1) = 1.
        DO 20 J=1,K
          DO 10 I=1,J
            HH(I) = H(I)
   10     CONTINUE
          H(1) = 0.
          DO 20 I=1,J
            LI = L+I
            LJ = LI-J
            F = HH(I)/(T(LI)-T(LJ))
            H(I) = H(I)+F*(T(LI)-X)
            H(I+1) = F*(X-T(LJ))
   20   CONTINUE
        RETURN
```

```
      END
      SUBROUTINE COSSIN(PIV,wI,ww,COS,SIN)
C   SUBROUTINE COSSIN CALCULATES THE PARAMETERS OF A GIVENS
C   TRANSFORMATION WITHOUT SQUARE ROOTS.
      STORE = PIV*wI
      DD = ww+STORE*PIV
      IF(ABS(DD).LT.1.E-36) DD=1.E-36
      COS = ww/DD
      SIN = STORE/DD
      ww = DD
      wI = COS*wI
      RETURN
      END
      SUBROUTINE ROTATE(PIV,COS,SIN,A,B)
C   SUBROUTINE ROTATE APPLIES A GIVENS ROTATION TO A AND B.
      STORE = B
      B = COS*STORE+SIN*A
      A = A-PIV*STORE
      RETURN
      END
      SUBROUTINE BACK(A,Z,N,K,C)
C   SUBROUTINE BACK CALCULATES THE SOLUTION OF THE SYSTEM OF
C   EQUATIONS A*C = Z WITH A A N X N UNIT UPPER TRIANGULAR MATRIX
C   OF BANDWIDTH K+1.
C   ATTENTION: THE FIRST DIMENSION SPECIFICATION OF MATRIX A MUST
C   BE THE SAME AS IN THE CALLING PROGRAM.
      DIMENSION A(1000,K),Z(N),C(N)
      C(N) = Z(N)
      I = N-1
      IF(I.EQ.0) GO TO 30
      DO 20 J=2,N
        STORE = Z(I)
        I1 = K
        IF(J.LE.K) I1 = J-1
        M = I
        DO 10 L=1,I1
          M = M+1
          STORE = STORE-C(M)*A(I,L)
10      CONTINUE
        C(I) = STORE
        I = I-1
20    CONTINUE
30    RETURN
      END
      SUBROUTINE NKNOT(X,M,T,N,FPINT,NRDATA,NRINT)
C   SUBROUTINE NKNOT LOCATES AN ADDITIONAL KNOT FOR A SPLINE OF DEGREE
C   K AND ADJUSTS THE CORRESPONDING PARAMETERS,I.E.
C     T     : THE POSITION OF THE KNOTS.
```

```
C     N      : THE NUMBER OF KNOTS.
C     NRINT  : THE NUMBER OF KNOTINTERVALS.
C     FPINT  : THE SUM OF SQUARES OF RESIDUAL RIGHT HAND SIDES
C              FOR EACH KNOT INTERVAL.
C     NRDATA: THE NUMBER OF DATA POINTS INSIDE EACH KNOT INTERVAL.
C   THE ARRAYS T,FPINT AND NRDATA MUST HAVE THE SAME DIMENSION
C   SPECIFICATIONS AS IN THE CALLING PROGRAM.
      DIMENSION X(M),T(1000),FPINT(1000),NRDATA(1000)
      K = (N-NRINT-1)/2
C   SEARCH FOR KNOT INTERVAL T(NUMBER+K) <= X <= T(NUMBER+K+1) WHERE
C   FPINT(NUMBER) IS MAXIMAL ON THE CONDITION THAT NRDATA(NUMBER)
C   NOT EQUALS ZERO.
      FPMAX = 0.
      JBEGIN = 1
      DO 20 J=1,NRINT
        JPOINT = NRDATA(J)
        IF(FPMAX.GE.FPINT(J) .OR. JPOINT.EQ.0) GO TO 10
        FPMAX = FPINT(J)
        NUMBER = J
        MAXPT = JPOINT
        MAXBEG = JBEGIN
   10   JBEGIN = JBEGIN+JPOINT+1
   20 CONTINUE
C   LET COINCIDE THE NEW KNOT T(NUMBER+K+1) WITH A DATA POINT X(NRX)
C   INSIDE THE OLD KNOT INTERVAL T(NUMBER+K) <= X <= T(NUMBER+K+1).
      IHALF = MAXPT/2+1
      NRX = MAXBEG+IHALF
      NEXT = NUMBER+1
      IF(NEXT.GT.NRINT) GO TO 40
C   ADJUSTS THE DIFFERENT PARAMETERS.
      DO 30 J=NEXT,NRINT
        JJ = NEXT+NRINT-J
        FPINT(JJ+1) = FPINT(JJ)
        NRDATA(JJ+1) = NRDATA(JJ)
        JK = JJ+K
        T(JK+1) = T(JK)
   30 CONTINUE
   40 NRDATA(NUMBER) = IHALF-1
      NRDATA(NEXT) = MAXPT-IHALF
      FPINT(NUMBER) = FPMAX*FLOAT(NRDATA(NUMBER))/FLOAT(MAXPT)
      FPINT(NEXT) = FPMAX*FLOAT(NRDATA(NEXT))/FLOAT(MAXPT)
      JK = NEXT+K
      T(JK) = X(NRX)
      N = N+1
      NRINT = NRINT+1
      RETURN
      END
      SUBROUTINE DISCU(T,N,K2,B)
```

```
C  SUBROUTINE DISCO CALCULATES THE DISCONTINUITY JUMPS OF THE KTH
C  DERIVATIVE OF THE B-SPLINES OF DEGREE K AT THE KNOTS T(K+2)..T(N-K-1)
C  THE FIRST DIMENSION SPECIFICATION OF THE MATRIX B MUST BE THE SAME AS
C  IN THE CALLING PROGRAM; H MUST HAVE A DIMENSION SPECIFICATION AT
C  LEAST 2*K+2.
      DIMENSION T(N),B(1000,K2),H(12)
      K1 = K2-1
      K = K1-1
      NK1 = N-K1
      DO 40 L=K2,NK1
        LMK = L-K1
        DO 10 J=1,K1
          IK = J+K1
          LJ = L+J
          LK = LJ-K2
          H(J) = T(L)-T(LK)
          H(IK) = T(L)-T(LJ)
10      CONTINUE
        LP = LMK
        DO 30 J=1,K2
          JK = J+K
          PROD = 1.
          DO 20 I=J,JK
            PROD = PROD*H(I)
20        CONTINUE
          LK = LP+K1
          B(LMK,J) = (T(LK)-T(LP))/PROD
          LP = LP+1
30      CONTINUE
40    CONTINUE
      RETURN
      END
      FUNCTION RATION(P1,F1,P2,F2,P3,F3)
C  GIVEN THREE POINTS (P1,F1),(P2,F2) AND (P3,F3), FUNCTION RATION
C  GIVES THE VALUE OF P SUCH THAT THE RATIONAL INTERPOLATING FUNCTION
C  OF THE FORM R(P) = (U*P+V)/(P+W) EQUALS ZERO AT P.
      IF(P3.GT.0.) GO TO 10
C  VALUE OF P IN CASE P3 = INFINITY.
      P = (P1*(F1-F3)*F2-P2*(F2-F3)*F1)/((F1-F2)*F3)
      GO TO 20
C  VALUE OF P IN CASE P3 /= INFINITY.
10    H1 = F1*(F2-F3)
      H2 = F2*(F3-F1)
      H3 = F3*(F1-F2)
      P = -(P1*P2*H3+P2*P3*H1+P3*P1*H2)/(P1*H1+P2*H2+P3*H3)
C  ADJUST THE VALUE OF P1,F1,P3 AND F3 SUCH THAT F1 > 0 AND F3 < 0.
20    IF(F2.LT.0.) GO TO 30
      P1 = P2
```

```
      F1 = F2
      GO TO 40
   30 P3 = P2
      F3 = F2
   40 RATION = P
      RETURN
      END
      FUNCTION DERIV(T,N,C,NK1,NU,ARG,L)
C  FUNCTION DERIV EVALUATES A SPLINE S(X) OF DEGREE K WHICH IS
C  GIVEN IN ITS NORMALIZED B-SPLINE REPRESENTATION OR CALCULATES
C  DERIVATIVES OF ANY SPECIFIED ORDER NU.
C
C  CALLING SEQUENCE
C     VALUE = DERIV(T,N,C,NK1,NU,ARG,L)
C
C  INPUT PARAMETERS:
C    T    : ARRAY,MINIMUM LENGTH N, WHICH CONTAINS THE POSITION
C           OF THE KNOTS OF S(X),I.E. THE POSITION OF THE INTERIOR
C           KNOTS T(K+2),...T(N-K-1) AS WELL AS THE POSITION OF THE
C           KNOTS T(1),...T(K+1) AND T(N-K),...T(N) WHICH ARE NEEDED
C           FOR THE B-SPLINE REPRESENTATION.
C    N    : INTEGER VALUE GIVING THE TOTAL NUMBER OF KNOTS OF S(X).
C    C    : ARRAY,LENGTH NK1, CONTAINING THE B-SPLINE COEFFICIENTS.
C    NK1  : INTEGER VALUE,GIVING THE DIMENSION OF S(X),I.E.NK1 = N-K-1.
C    NU   : INTEGER VALUE WHICH GIVES THE ORDER OF THE DERIVATIVE.
C    ARG  : REAL VALUE,GIVING THE VALUE OF THE ARGUMENT.
C    L    : INTEGER VALUE WHICH SPECIFIES THE POSITION OF THE ARGUMENT
C           I.E.   T(L) <= ARG <  T(L+1)   OR
C                  L = NK1 IF ARG = T(NK1+1).
C
C  OUTPUT PARAMETER:
C    VALUE: REAL VALUE,GIVING THE VALUE OF THE NUTH DERIVATIVE OF
C           S(X) AT X = ARG.
C
C  OTHER SUBROUTINES REQUIRED: NONE.
C
C  RESTRICTIONS:
C    1) NU >= 0
C    2) T(K+1) <= ARG <= T(NK1+1)
C  THE DIMENSION SPECIFICATION OF THE ARRAY H MUST BE AT LEAST K+1.
      DIMENSION T(N),C(NK1),H(6)
      DERIV = 0.
      K1 = N-NK1
      IF(NU.LT.0 .OR. NU.GE.K1) RETURN
      DO 100 I=1,K1
        IK = L+I-K1
        H(I) = C(IK)
  100 CONTINUE
```

```
      IF(NU.EQ.0) GO TO 300
      NU1 = NL+1
      DO 200 J=2,NU1
        DO 200 JJ=J,K1
          I  = J+K1-JJ
          LI = L+I-K1
          LJ = L+I-J+1
          H(I) = (H(I)-H(I-1))/(T(LJ)-T(LI))
200   CONTINUE
      IF(NU.EQ.K1-1) GO TO 500
300   NU2 = NL+2
      DO 400 J=NU2,K1
        DO 400 JJ=J,K1
          I  = J+K1-JJ
          LI = L+I-K1
          LJ = L+I-J+1
          H(I) = ((ARG-T(LI))*H(I)+(T(LJ)-ARG)*H(I-1))/(T(LJ)-T(LI))
400   CONTINUE
500   DERIV = H(K1)
      IF(NU.EQ.0) RETURN
      DO 600 I=1,NU
        DERIV = DERIV*FLOAT(K1-I)
600   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PERCLO(X,Y,Z,W,M,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
C   GIVEN THE SET OF DATA POINTS (X(I),Y(I)),I=1,2,...M SUCH THAT
C   X(M) = X(1) AND Y(M) = Y(1). GIVEN ALSO THE CORRESPONDING PARAMETER
C   VALUES Z(I) AND THE WEIGHTS W(I),I=1,2,...M, PERCLO FINDS A SMOOTH,
C   CLOSED APPROXIMATING CURVE WITH PARAMETER REPRESENTATION X = SX(Z),
C   Y = SY(Z).
C   SX(Z) AND SY(Z) ARE TWO PERIODIC SPLINES ( PERIOD PER = Z(M)-Z(1) )
C   OF DEGREE K WITH THE NUMBER AND THE POSITION OF THE KNOTS T(J),J=
C   1,2,...N AUTOMATICALLY CHOSEN BY THE ROUTINE.
C   THE SMOOTHNESS OF SX(Z) AND SY(Z) IS ACHIEVED BY MINIMALIZING THE
C   SUM(DX(R)**2+DY(R)**2) WHERE DX(R) AND DY(R) STAND FOR THE DISCON-
C   TINUITY JUMP OF THE KTH DERIVATIVE OF SX(Z) AND SY(Z) AT T(R),R=
C   K+2,...N-K-1. THE AMOUNT OF SMOOTHNESS IS DETERMINED BY THE CONDITION
C   THAT F(P) = SUM(W(I)*((X(I)-SX(Z(I)))**2+(Y(I)-SY(Z(I)))**2)),I=1,...M-1 BE <=S
C   WITH S A GIVEN NON-NEGATIVE CONSTANT.
C   THE SPLINE FUNCTIONS SX(Z) AND SY(Z) ARE GIVEN IN THEIR B-SPLINE
C   REPRESENTATION (B-SPLINE COEFFICIENTS CX(J),RESP. CY(J),J=1,...N-K-1)
C   AND CAN BE EVALUATED BY MEANS OF FUNCTION DERIV.
C
C   CALLING SEQUENCE:
C      CALL PERCLO(X,Y,Z,W,M,K,S,N,T,CX,CY,FP,IOPT,IPAR,IER)
C
C   INPUT PARAMETERS:
C     X    : ARRAY,LENGTH M, CONTAINING THE ABSCISSAE OF THE DATA POINTS;
C            IT IS PRESUMED THAT X(M) = X(1).
C     Y    : ARRAY,LENGTH M, CONTAINING THE ORDINATES OF THE DATA POINTS;
C            IT IS PRESUMED THAT Y(M) = Y(1).
C     W    : ARRAY,LENGTH M,CONTAINING THE WEIGHTS W(I);
C            W(M) MAY BE CHOSEN ARBITRARILY.
C     M    : INTEGER VALUE ,CONTAINING THE NUMBER OF DATA POINTS.
C     K    : INTEGER VALUE,CONTAINING THE DEGREE OF SX(Z) AND SY(Z).
C     S    : REAL VALUE,CONTAINING THE SMOOTHING FACTOR.
C     IOPT: INTEGER VALUE WHICH TAKES THE VALUE 0 OR 1.
C       IOPT=0: THE ROUTINE WILL RESTART ALL COMPUTATIONS.
C       IOPT=1: THE ROUTINE WILL START WITH THE KNOTS FOUND AT THE
C               LAST CALL OF THE ROUTINE. IF IOPT=1 THE OUTPUT
C               PARAMETERS T AND N ARE INPUT PARAMETERS AS WELL.
C               IF IOPT=1 THE USER MUST PROVIDE WITH A COMMON BLOCK
C                  COMMON/OPT1/NRDATA(NEST),FP0,FPOLD,NPLUS
C     IPAR : INTEGER FLAG.
C       IPAR = 0: FOR EACH DATA POINT (X(I),Y(I)) THE PROGRAM AUTOMATICALLY
C                 CHOOSES A CORRESPONDING VALUE OF THE PARAMETER Z, I.E.
C                 Z(1)=0; Z(I)=Z(I-1)+SQRT((X(I)-X(I-1))**2+(Y(I)-Y(I-1))**2)
C       IPAR = 1: THE USER HIMSELF PROVIDES WITH THE VALUES OF THE
C                 PARAMETER Z.
C     Z    : ARRAY,LENGTH M, CONTAINING THE VALUES OF THE PARAMETER Z
C            (IPAR = 1)
C
```

```
C   OUTPUT PARAMETERS:
C     T    : ARRAY,LENGTH NEST (SEE DATA INITIALIZATION STATEMENT),
C            WHICH CONTAINS THE POSITION OF THE KNOTS,I.E. THE POSITION
C            OF THE INTERIOR KNOTS T(K+2),...T(N-K-1), AS WELL AS THE
C            POSITION OF THE KNOTS T(1),T(2),...T(K+1) AND T(N-K),...T(N),
C            WHICH ARE NEEDED FOR THE B-SPLINE REPRESENTATION OF SX(Z)
C            AND SY(Z).
C     CX   : ARRAY,LENGTH NEST,CONTAINING THE B-SPLINE COEFF. OF SX(Z).
C     CY   : ARRAY,LENGTH NEST,CONTAINING THE B-SPLINE COEFF. OF SY(Z).
C     N    : INTEGER VALUE,CONTAINING THE TOTAL NUMBER OF KNOTS.
C     FP   : REAL VALUE,WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2) +
C            SUM(WI*(YI-SY(ZI))**2), I=1,2,...M-1.
C     IER  : ERROR CODE
C       IER=0: NORMAL RETURN.
C       IER=-1:NORMAL RETURN,SX(Z) AND SY(Z) ARE INTERPOLATING
C              PERIODIC SPLINES.
C       IER=-2:NORMAL RETURN,SX(Z) AND SY(Z) ARE CONSTANT FUNCTIONS.
C       IER>0 :ABNORMAL TERMINATION.
C         IER=1:THE REQUIRED STORAGE SPACE EXCEEDS THE AVAILABLE
C               STORAGE SPACE,SPECIFIED BY THE PARAMETER NEST.
C               PROBABLY CAUSES:NEST OR S TOO SMALL.
C         IER=2:A THEORETICALLY IMPOSSIBLE RESULT WAS FOUND DURING
C               THE ITERATION PROCESS.
C               PROBABLY CAUSES:TOL TOO SMALL
C         IER=3:THE MAXIMAL NUMBER OF ITERATIONS MAXIT HAS BEEN
C               REACHED.
C               PROBABLY CAUSES:MAXIT OR TOL TOO SMALL.
C         IER=10:SOME OF THE INPUT DATA ARE INVALID(SEE RESTRICTIONS).
C
C   RESTRICTIONS:
C     1) M > 1.
C     2) K > 0.
C     3) Z(R) < Z(R+1), R=1,2,...M-1.    (IPAR=1)
C     4) W(R) > 0, R=1,2,...M-1.
C     5) S >= 0.
C     6) NEST >= 2*K+2.
C
C   OTHER SUBROUTINES REQUIRED:
C     BSPLIN,CCSSIN,ROTATE,PBACK,NKNOT,DISCO AND RATION.
      DIMENSION X(M),Y(M),W(M),Z(M),T(1000),CX(1000),CY(1000),
     < NRDATA(1000),FPINT(1000),RX(1000),RY(1000),DIAG(1000),
     < A1(1000,5),A2(1000,5),G1(1000,6),G2(1000,6),B(1000,7),
     < Q(2000,6),H(6),H1(7),H2(6),DPRIME(1000)
C   COMMON/OPT1/NRDATA(NEST),FP0,FPOLD,NPLUS
C     NRDATA: INTEGER ARRAY,LENGTH NEST,WHICH GIVES THE NUMBER OF
C             DATA POINTS INSIDE EACH KNOT INTERVAL.
C     FP0   : REAL VALUE,WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2) +
C             SUM(WI*(YI-SY(ZI))**2), I=1,2,...M-1 FOR SX(Z) AND SY(Z)
```

```
C              BEING THE LEAST-SQUARES POLYNOMIALS OF DEGREE 0.
C     FPOLD :  REAL VALUE,WHICH CONTAINS THE SUM(WI*(XI-SX(ZI))**2)+
C              SUM(WI*(SY(ZI)-YI)**2),I=1,2,...M-1, FOR SX(Z) AND SY(Z)
C              BEING THE LEAST-SQUARES PERIODIC SPLINES CORRESPONDING
C              TO THE LAST FOUND SET OF KNOTS BUT ONE.
C     NPLUS :  INTEGER VALUE,GIVING THE NUMBER OF KNOTS OF THE LAST
C              SET MINUS THE NUMBER OF THE LAST SET BUT ONE.
      COMMON/CPT1/NRDATA,FPO,FPOLD,NPLUS
C   DATA INITIALIZATION STATEMENT TO SPECIFY
C     TOL    : THE REQUESTED RELATIVE ACCURACY FOR THE ROOT OF F(P) = S.
C     MAXIT: THE MAXIMAL NUMBER OF ITERATIONS ALLOWED.
C     NEST :  AN OVER-ESTIMATE OF THE NUMBER OF KNOTS N. THIS PARAMETER
C             MUST BE SET BY THE USER TO INDICATE THE STORAGE SPACE
C             AVAILABLE TO THE SUBROUTINE. THE DIMENSION SPECIFICATIONS
C             OF THE ARRAYS T,CX,CY,NRDATA,FPINT,RX,RY,DIAG,DPRIME(N),
C             A1(N,K),A2(N,K),G1(N,K+1),G2(N,K+1),B(N,K+2),Q(M,K+1),
C             H(K+1),H1(K+2) AND H2(K+1) DEPEND ON N,M AND K. SINCE N IS
C             UNKNOWN AT THE TIME THE USER SETS UP THE DIMENSION INFORMATION
C             AN OVER-ESTIMATE OF THESE ARRAYS WILL GENERALLY BE MADE.
C             THE FOLLOWING REMARKS ARE INTENDED TO HELP THE USER
C               1) 2*K+2 <= N <= M+2*K.
C               2) THE SMALLER THE VALUE OF S, THE GREATER N WILL BE.
C               3) NORMALLY N = M/2 IS AN OVER-ESTIMATE.
      DATA TOL/0.001/,MAXIT/20/,NEST/1000/
C   BEFORE STARTING COMPUTATIONS A DATA CHECK IS MADE. IF THE INPUT
C   DATA ARE INVALID CONTROLE IS IMMEDIATELY REPASSED TO THE DRIVER
C   PROGRAM (IER=10).
      IER = 0
      K1 = K+1
      K2 = K1+1
      K3 = 3*K+1
      NMIN = 2*K1
      IF(K.LE.0) IER = 10
      IF(M.LT.2 .OR. NEST.LT.NMIN) IER = 10
      IF(S.LT.0.) IER = 10
      M1 = M-1
      IF(IER.NE.0) GO TO 660
C   CHECK WHETHER THE Z-VALUES ARE PROVIDED WITH BY THE USER.
      IF(IPAR.NE.0) GO TO 6
C   FIND FOR EACH DATA POINT A CORRESPONDING VALUE OF THE PARAMETER Z
      Z(1) = 0.
      DO 4 I=2,M1
        Z(I) = Z(I-1)+SQRT((X(I)-X(I-1))**2+(Y(I)-Y(I-1))**2)
    4 CONTINUE
      Z(M) = Z(M1)+SQRT((X(1)-X(M1))**2+(Y(1)-Y(M1))**2)
    6 DO 10 I=1,M1
        IF(Z(I).GE.Z(I+1) .OR. W(I).LE.0.) IER = 10
   10 CONTINUE
```

```
      IF(IER.NE.0) GO TO 660
C  CALCULATION OF ACC, THE ABSOLUTE TOLERANCE FOR THE ROOT OF F(P)=S.
      ACC = TOL*S
C  DETERMINE THE LENGTH OF THE PERIOD OF SX(Z) AND SY(Z).
      PER = Z(M)-Z(1)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  PART 1: DETERMINATION OF THE NUMBER OF KNOTS AND THEIR POSITION     C
C  ****************************************************************     C
C  GIVEN A SET OF KNOTS WE COMPUTE THE LEAST-SQUARES PERIODIC SPLINES  C
C  SXINF(Z) AND SYINF(Z).IF THE SUM F(P=INF)<=S WE ACCEPT THE CHOICE   C
C  OF KNOTS.THE INITIAL CHOICE DEPENDS ON THE VALUE OF S AND IOPT.     C
C    IF S=0 WE HAVE SPLINE INTERPOLATION; IN THAT CASE THE NUMBER OF   C
C    KNOTS EQUALS NMAX = M+2*K.                                        C
C    IF S > 0 AND                                                      C
C      IOPT=0 WE FIRST COMPUTE THE LEAST-SQUARES POLYNOMIALS OF        C
C      DEGREE K; N=NMIN= 2*K+2. SINCE SX(Z) AND SY(Z) MUST BE PERIODIC C
C      WE FIND THAT THEY ARE CONSTANT FUNCTIONS.                       C
C      IOPT=1 WE START WITH THE SET OF KNOTS FOUND AT THE LAST         C
C      CALL OF THE ROUTINE, EXCEPT FOR THE CASE THAT S > FP0; THEN     C
C      WE COMPUTE DIRECTLY THE LEAST-SQUARES PERIODIC POLYNOMIALS.     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C  DETERMINE NMAX, THE NUMBER OF KNOTS FOR PERIODIC SPLINE INTERPOLATION
      NMAX = M+2*K
      IF(S.GT.0. .OR. NMAX.EQ.NMIN) GO TO 30
C  IF S=0, SX(Z) AND SY(Z) ARE INTERPOLATING SPLINES.
      N = NMAX
C  TEST WHETHER THE REQUIRED STORAGE SPACE EXCEEDS THE AVAILABLE ONE.
      IF(N.GT.NEST) GO TO 620
C  FIND THE POSITION OF THE INTERIOR KNOTS IN CASE OF INTERPOLATION.
      DO 20 I=2,M1
        J = I+K
        T(J) = Z(I)
   20 CONTINUE
      GO TO 50
C  IF S > 0 OUR INITIAL CHOICE DEPEND ON THE VALUE OF IOPT.
C  IF IOPT=0 OR IOPT=1 AND S>=FP0, WE START COMPUTING THE LEAST-SQUARES
C  PERIODIC POLYNOMIALS (I.E. CONSTANT FUNCTIONS).
C  IF IOPT=1 AND FP0>S WE START COMPUTING THE LEAST-SQUARES PERIODIC
C  SPLINES ACCORDING TO THE SET OF KNOTS FOUND AT THE LAST CALL OF THE
C  ROUTINE.
   30 IF(IOPT.LE.0) GO TO 35
      IF(N.GT.NMIN .AND. S.LT.FP0) GO TO 50
C  THE CASE THAT SX(Z) AND SY(Z) ARE CONSTANT FUNCTIONS IS TREATED
C  SEPARETELY. WE DETERMINE THE LEAST-SQUARES COMSTANT FUNCTIONS
C  C1 AND C2 AND COMPUTE FP0 AT THE SAME TIME.
   35 FP0 = 0.
      D1 = 0.
      C1 = 0.
```

```
      C2 = 0.
      DO 40 II=1,M1
        WI = W(II)
        XI = X(II)
        YI = Y(II)
        CALL CCSSIN(1.,WI,D1,COS,SIN)
        CALL ROTATE(1.,COS,SIN,XI,C1)
        CALL ROTATE(1.,COS,SIN,YI,C2)
        FP0 = FP0+WI*(XI**2+YI**2)
 40   CONTINUE
C  TEST WHETHER THE APPROXIMATION X=C1,Y=C2 IS A SOLUTION OF OUR PROBLEM
      FPMS = FP0-S
      IF(FPMS.LT.ACC .OR. NMAX.EQ.NMIN) GO TO 640
C  START COMPUTING THE LEAST-SQUARES PERIODIC SPLINES WITH ONE
C  INTERIOR KNOT.
      FPOLD = FP0
      NPLUS = 1
      N = NMIN+1
C  TEST WHETHER THE REQUIRED STORAGE SPACE EXCEEDS THE AVAILABLE ONE.
      IF(N.GT.NEST) GO TO 620
      MM = (M+1)/2
      T(K2) = Z(MM)
      NRDATA(1) = MM-2
      NRDATA(2) = M1-MM
C  MAIN LOOP FOR THE DIFFERENT SETS OF KNOTS. M IS A SAVE UPPER
C  BOUND FOR THE NUMBER OF TRIALS.
 50   DO 340 ITER=1,M
C  FIND NRINT, THE NUMBER OF KNOT INTERVALS.
      NRINT = N-NMIN+1
C  FIND THE POSITION OF THE ADDITIONAL KNOTS WHICH ARE NEEDED FOR
C  THE B-SPLINE REPRESENTATION OF SX(Z) AND SY(Z). IF WE TAKE
C      T(K+1) = Z(1), T(N-K) = Z(M)
C      T(K+1-J) = T(N-K-J) - PER, J=1,2,...K
C      T(N-K+J) = T(K+1+J) + PER, J=1,2,...K
C  THEN SX(Z) AND SY(Z) ARE PERIODIC SPLINES IF THE B-SPLINE
C  COEFFICIENTS SATISFY THE FOLLOWING CONDITIONS
C      CX(J) = CX(N7+J),J=1,...K  (**)   WITH N7 = N-2*K-1.
C      CY(J) = CY(N7+J)
      T(K1) = Z(1)
      NK1 = N-K1
      NK2 = NK1+1
      T(NK2) = Z(M)
      DO 60 J=1,K
        I1 = NK2+J
        I2 = NK2-J
        J1 = K1+J
        J2 = K1-J
        T(I1) = T(J1)+PER
```

```
          T(J2) = T(I2)-PER
   60     CONTINUE
C  COMPUTE THE B-SPLINE COEFFICIENTS CX(J) AND CY(J),J=1,..N7 OF THE LEAST-
C  SQUARES PERIODIC SPLINES SXINF(Z) AND SYINF(Z). THE OBSERVATION MATRIX
C  A IS BUILT UP ROW BY ROW WHILE TAKING ACCOUNT OF CONDITION (**) AND IS
C  REDUCED TO UPPER TRIANGULAR FORM BY GIVENS TRANSFORMATIONS WITHOUT
C  SQUARE ROOTS. AT THE SAME TIME FP=F(P=INF) IS COMPUTED.
C  THE N7 X N7 TRIANGULARISED UNIT UPPER MATRIX A HAS THE FORM
C                A1 '
C         A =     ' A2
C                0  '
C  WITH A2 A N7 X K MATRIX AND A1 A N10 X N10 UNIT UPPER TRIANGULAR
C  MATRIX OF BANDWITH K+1 ( N10 = N7-K).
C  INITIALIZATION.
          DO 70 I=1,NK1
            RX(I) = 0.
            RY(I) = 0.
            DIAG(I) = 0.
            DO 70 J=1,K
              A1(I,J) = 0.
   70     CONTINUE
          N10 = N-K3
          N7 = N10+K
          JPER = 0
          FP = 0.
          L = K1
          DO 250 IT=1,M1
C  FETCH THE CURRENT DATA POINT X(IT),Y(IT),Z(IT)
            XI = X(IT)
            YI = Y(IT)
            WI = W(IT)
            ZI = Z(IT)
C  SEARCH FOR KNOT INTERVAL T(L) <= ZI < T(L+1).
            IF(ZI.GE.T(L+1)) L = L+1
C  EVALUATE THE (K+1) NON-ZERO B-SPLINES AT ZI AND STORE THEM IN Q.
            CALL BSPLIN(T,N,K,ZI,L,H)
            DO 75 I=1,K1
              IF(H(I).LT.0.1E-07) H(I) = 0.
              Q(IT,I) = H(I)
   75       CONTINUE
C  TEST WHETHER THE B-SPLINES NJ,K+1(Z),J=1+N7,...NK1 ARE ALL ZERO AT ZI
            L5 = L-K1
            IF(L5.LT.N10) GO TO 285
            IF(JPER.NE.0) GO TO 160
C  INITIALIZE THE MATRIX A2.
            DO 80 I=1,N7
              DO 80 J=1,K
                A2(I,J) = 0.
```

```
   80       CONTINUE
            JK = N10
            DO 110 I=1,K
               IK = JK
               DO 90 J=1,K
                  IF(IK.LE.0) GO TO 100
                  A2(IK,I) = A1(IK,J)
                  IK = IK-1
   90          CONTINUE
  100          JK = JK+1
  110       CONTINUE
            JPER = 1
C   IF ONE OF THE B-SPLINES NJ,K+1(Z),J=N7+1,...NK1 IS NOT ZERO AT ZI
C   WE TAKE ACCOUNT OF CONDITION (**) FOR SETTING UP THE NEW ROW
C   OF THE OBSERVATION MATRIX A. THIS ROW IS STORED IN THE ARRAYS H1
C   (THE PART WITH RESPECT TO A1) AND H2 (THE PART WITH
C   RESPECT TO A2).
  160       DO 170 I=1,K
               H1(I) = 0.
               H2(I) = 0.
  170       CONTINUE
            H1(K1) = 0.
            J = L5-N10
            DO 210 I=1,K1
               J = J+1
               L0 = J
  180          L1 = L0-K
               IF(L1.LE.0) GO TO 200
               IF(L1.LE.N10) GO TO 190
               L0 = L1-N10
               GO TO 180
  190          H1(L1) = H(I)
               GO TO 210
  200          H2(L0) = H2(L0)+H(I)
  210       CONTINUE
C   ROTATE THE NEW ROW OF THE OBSERVATION MATRIX INTO TRIANGLE
C   BY GIVENS TRANSFORMATIONS WITHOUT SQUARE ROOTS.
            IF(N10.LE.0) GO TO 250
C   ROTATION WITH THE ROWS 1,2,...N10 OF MATRIX A.
            DO 240 J=1,N10
               PIV = H1(1)
C   CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
               CALL COSSIN(PIV,W1,DIAG(J),COS,SIN)
C   TRANSFORMATION TO THE RIGHT HAND SIDES.
               CALL ROTATE(PIV,COS,SIN,XI,RX(J))
               CALL ROTATE(PIV,COS,SIN,YI,RY(J))
C   TRANSFORMATIONS TO THE LEFT HAND SIDE WITH RESPECT TO A2.
               DO 220 I=1,K
```

```
                CALL ROTATE(PIV,COS,SIN,H2(I),A2(J,I))
  220         CONTINUE
              IF(J.EQ.N10) GO TO 250
              I2 = MIN0(N10-J,K)
C   TRANSFORMATIONS TO THE LEFT HAND SIDE WITH RESPECT TO A1.
              DO 230 I=1,I2
                CALL ROTATE(PIV,COS,SIN,H1(I+1),A1(J,I))
                H1(I) = H1(I+1)
  230         CONTINUE
              H1(I2+1) = 0.
  240       CONTINUE
C   ROTATION WITH THE ROWS N10+1,...N7 OF MATRIX A.
  250       DO 270 J=1,K
              IF(WI.EQ.0.) GO TO 290
              IJ = N10+J
              IF(IJ.LE.0) GO TO 270
              PIV = H2(J)
C   CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
              CALL COSSIN(PIV,WI,DIAG(IJ),COS,SIN)
C   TRANSFORMATIONS TO RIGHT HAND SIDES.
              CALL ROTATE(PIV,COS,SIN,XI,RX(IJ))
              CALL ROTATE(PIV,COS,SIN,YI,RY(IJ))
              IF(J.EQ.K) GO TO 280
              J1 = J+1
C   TRANSFORMATIONS TO LEFT HAND SIDE.
              DO 260 I=J1,K
                CALL ROTATE(PIV,COS,SIN,H2(I),A2(IJ,I))
  260         CONTINUE
  270       CONTINUE
C   ADD CONTRIBUTION OF THIS ROW TO THE SUM OF SQUARES OF RESIDUAL
C   RIGHT HAND SIDES.
  280       FP = FP+WI**(XI**2+YI**2)
            GO TO 290
C   ROTATION OF THE NEW ROW OF THE OBSERVATION MATRIX INTO
C   TRIANGLE IN CASE THE B-SPLINES NJ,K+1(Z),J=N7+1,...N-K-1 ARE ALL ZERO
C   AT ZI.
  285       J = L5
            DO 140 I=1,K1
              IF(WI.EQ.0.) GO TO 290
              J = J+1
              PIV = H(I)
              IF(PIV.EQ.0.) GO TO 140
C   CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
              CALL COSSIN(PIV,WI,DIAG(J),COS,SIN)
C   TRANSFORMATIONS TO RIGHT HAND SIDES.
              CALL ROTATE(PIV,COS,SIN,XI,RX(J))
              CALL ROTATE(PIV,COS,SIN,YI,RY(J))
              IF(I.EQ.K1) GO TO 150
```

```
                I2 = 0
                I3 = I+1
C   TRANSFORMATIONS TO LEFT HAND SIDE.
                DO 130 I1=I3,K1
                  I2 = I2+1
                  CALL ROTATE(PIV,COS,SIN,H(I1),A1(J,I2))
 130            CONTINUE
 140          CONTINUE
C   ADD CONTRIBUTION OF THIS ROW TO THE SUM OF SQUARES OF RESIDUAL
C   RIGHT HAND SIDES.
 150          FP = FP+WI*(XI**2+YI**2)
 290        CONTINUE
C   BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFFICIENTS CX(J),J=1,..N7
            CALL FBACK(A1,A2,RX,N7,K,CX)
C   BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFFICIENTS CY(J),J=1,..N7
            CALL FBACK(A1,A2,RY,N7,K,CY)
C   CALCULATE FROM CONDITION (**) THE COEFFICIENTS CX(N7+J) AND
C   CY(N7+J),J=1,2,...K.
            DO 295 I=1,K
              J = I+N7
              CX(J) = CX(I)
              CY(J) = CY(I)
 295        CONTINUE
C   TEST WHETHER THE APPROXIMATION X=SXINF(Z),Y=SYINF(Z) IS AN
C   ACCEPTABLE SOLUTION.
            FPMS = FP-S
            IF(ABS(FPMS).LT.ACC) GO TO 660
C   IF F(P=INF) < S ACCEPT THE CHOICE OF KNOTS.
            IF(FPMS.LT.0.) GO TO 350
C   IF N=NMAX, SXINF(Z) AND SYINF(Z) ARE INTERPOLATING SPLINES.
            IF(N.EQ.NMAX) GO TO 630
C   INCREASE THE NUMBER OF KNOTS.
C   IF N=NEST WE CANNOT INCREASE THE NUMBER OF KNOTS BECAUSE OF THE
C   STORAGE CAPACITY LIMITATION.
            IF(N.EQ.NEST) GO TO 620
C   DETERMINE THE NUMBER OF KNOTS NPLUS WE ARE GOING TO ADD.
            NPL1 = NPLUS*2
            IF(FPOLD-FP.GT.ACC) NPL1 = FLOAT(NPLUS)*FPMS/(FPOLD-FP)
            NPLUS = MIN0(NPLUS*2,MAX0(NPL1,NPLUS/2,1))
            FPOLD = FP
C   COMPUTE THE SUM(WI*(YI-SXINF(ZI))**2)+SUM(WI*(YI-SYINF(ZI))**2) FOR
C   EACH INTERVAL T(J+K) <= ZI <= T(J+K+1) AND STORE IT IN FPINT(J)
C   J=1,2,...NRINT.
            FPART = 0.
            I = 1
            L = K1
            DO 320 IT=1,M1
              IF(Z(IT).LT.T(L)) GO TO 300
```

```
            NEW = 1
            L = L+1
  300       TERM1 = 0.
            TERM2 = 0.
            LO = L-K2
            DO 310 J=1,K1
              LC = LO+1
              TERM1 = TERM1+CX(LO)*Q(IT,J)
              TERM2 = TERM2+CY(LO)*Q(IT,J)
  310       CONTINUE
            TERM = W(IT)*((TERM1-X(IT))**2+(TERM2-Y(IT))**2)
            FPART = FPART+TERM
            IF(NEW.EQ.0) GO TO 320
            IF(L.GT.K2) GO TO 315
            FPINT(NRINT) = TERM
            NEW = 0
            GO TO 320
  315       STORE = TERM*0.5
            FPINT(I) = FPART-STORE
            I = I+1
            FPART = STORE
            NEW = 0
  320     CONTINUE
          FPINT(NRINT) = FPINT(NRINT)+FPART
          DO 330 L=1,NPLUS
C   ADD A NEW KNOT
            CALL NKNOT(Z,M,T,N,FPINT,NRDATA,NRINT)
C   TEST WHETHER WE CANNOT FURTHER INCREASE THE NUMBER OF KNOTS.
            IF(N.EQ.NMAX .OR. N.EQ.NEST) GO TO 340
  330     CONTINUE
C   RESTART THE COMPUTATIONS WITH THE NEW SET OF KNOTS.
  340   CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   PART 2:DETERMINATION OF THE SMOOTHING PERIODIC SPLINES SXP(Z),SYP(Z)C
C   **********************************************************************C
C   WE HAVE DETERMINED THE NUMBER OF KNOTS AND THEIR POSITION.           C
C   WE NOW COMPUTE THE B-SPLINE COEFFICIENTS OF THE SMOOTHING SPLINES    C
C   SXP(Z) AND SYP(Z).THE OBSERVATION MATRIX IS EXTENDED BY THE ROWS OF C
C   MATRIX B EXPRESSING THAT THE KTH DERIVATIVE DISCONTINUITIES OF SXP(Z)C
C   AND SYP(Z) AT THE INTERIOR KNOTS T(K+2),..T(N-K-1) MUST BE ZERO.THE C
C   CORRESPONDING WEIGHTS OF THESE ADDITIONAL ROWS ARE SET TO 1/SQRT(P).C
C   ITERATIVELY WE THEN HAVE TO DETERMINE THE VALUE OF P SUCH THAT F(P) C
C   =SUM(W(I)*((XI-SXP(ZI))**2+(YI-SYP(ZI))**2)) BE=S.WE ALREADY KNOW THATC
C   THE LEAST-SQUARES CONSTANT FUNCTIONS CORRESPOND TO P=0, AND THAT     C
C   THE LEAST-SQUARES PERIODIC SPLINES CORRESPOND TO P=INFINITY. THE     C
C   ITERATION PROCESS WHICH IS PROPOSED HERE, MAKES USE OF RATIONAL      C
C   INTERPOLATION. SINCE F(P) IS A CONVEX AND STRICTLY DECREASING        C
C   FUNCTION OF P, IT CAN BE APPROXIMATED BY A RATIONAL FUNCTION         C
```

```
C   R(P) = (U*P+V)/(P+W). THREE VALUES OF P(P1,P2,P3) WITH CORRESPOND-   C
C   ING VALUES OF F(P) (F1=F(P1)-S,F2=F(P2)-S,F3=F(P3)-S) ARE USED       C
C   TO CALCULATE THE NEW VALUE OF P SUCH THAT R(P)=S. CONVERGENCE IS     C
C   GUARANTEED BY TAKING F1>0 AND F3<0.                                  C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C   EVALUATE THE DISCONTINUITY JUMP OF THE KTH DERIVATIVE OF THE
C   B-SPLINES AT THE KNOTS T(L),L=K+2,...N-K-1 AND STORE IN B.
 350  CALL DISCUT(T,N,K2,B)
C   INITIAL VALUE FOR P.
      P1 = 0.
      F1 = FP0-S
      P3 = -1.
      F3 = FPMS
      N11 = N10-1
      N8 = N7-1
      P = -F1/F3
      ICHECK = 0
C   ITERATION PROCESS TO FIND THE ROOT OF F(P) = S.
      DO 590 ITER=1,MAXIT
C   FORM THE MATRIX G  AS THE MATRIX A EXTENDED BY THE ROWS OF MATRIX B.
C   THE ROWS OF MATRIX B WITH WEIGHT 1/SQRT(P) ARE ROTATED INTO
C   THE TRIANGULARISED OBSERVATION MATRIX A.
C   AFTER TRIANGULARISATION OUR N7 X N7 MATRIX G TAKES THE FORM
C                G1 :
C         G =       : G2
C                0  :
C   WITH G2 A N7 X (K+1) MATRIX AND G1 A N11 X N11 UNIT UPPER TRIANGULAR
C   MATRIX OF BANDWIDTH K+2. ( N11 = N7-K-1)
      PINV = 1./P
C   STORE MATRIX A INTO G
      DO 360 I=1,N7
        DPRIME(I) = DIAG(I)
        CX(I) = RX(I)
        CY(I) = RY(I)
        G1(I,K1) = 0.
        G2(I,1) = 0.
        DO 360 J=1,K
          G1(I,J) = A1(I,J)
          G2(I,J+1) = A2(I,J)
 360    CONTINUE
      DO 370 J=1,K
        L = N10-J
        IF(L.LE.0) GO TO 375
        G2(L,1) = A1(L,J)
 370  CONTINUE
 375  DO 540 IT=1,N8
C   FETCH A NEW ROW OF MATRIX B AND STORE IT IN THE ARRAYS H1 (THE PART
C   WITH RESPECT TO G1) AND H2 (THE PART WITH RESPECT TO G2).
```

```
                XI = 0.
                YI = 0.
                WI = PINV
                DO 380 I=1,K1
                   H1(I) = 0.
                   H2(I) = 0.
  380           CONTINUE
                H1(K2) = 0.
                IF(IT.GT.N11) GO TO 420
                L = IT
                LO = IT
                DO 390 J=1,K2
                   IF(LO.EQ.N10) GO TO 400
                   H1(J) = B(IT,J)
                   LO = LO+1
  390           CONTINUE
                GO TO 470
  400           LO = 1
                DO 410 L1=J,K2
                   H2(LO) = B(IT,L1)
                   LO = LO+1
  410           CONTINUE
                GO TO 470
  420           L = 1
                I = IT-N10
                DO 460 J=1,K2
                   I = I+1
                   LO = I
  430              L1 = LO-K1
                   IF(L1.LE.0) GO TO 450
                   IF(L1.LE.N11) GO TO 440
                   LO = L1-N11
                   GO TO 430
  440              H1(L1) = B(IT,J)
                   GO TO 460
  450              H2(LO) = H2(LO)+B(IT,J)
  460           CONTINUE
                IF(N11.LE.0) GO TO 510
C  ROTATE THIS ROW INTO TRIANGLE BY GIVENS TRANSFORMATIONS WITHOUT
C  SQUARE ROOTS.
C  ROTATION WITH THE ROWS L,L+1,...N11.
  470           DO 500 J=L,N11
                   PIV = H1(1)
C  CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION.
                   CALL COSSIN(PIV,WI,DPRIME(J),COS,SIN)
C  TRANSFORMATION TO RIGHT HAND SIDES.
                   CALL ROTATE(PIV,COS,SIN,XI,CX(J))
                   CALL ROTATE(PIV,COS,SIN,YI,CY(J))
```

```
C  TRANSFORMATION TO THE LEFT HAND SIDE WITH RESPECT TO G2.
               DO 480 I=1,K1
                 CALL ROTATE(PIV,COS,SIN,H2(I),G2(J,I))
 480           CONTINUE
               IF(J.EQ.N11) GO TO 510
               I2 = MIN0(N11-J,K1)
C  TRANSFORMATION TO THE LEFT HAND SIDE WITH RESPECT TO G1.
               DO 490 I=1,I2
                 CALL ROTATE(PIV,COS,SIN,H1(I+1),G1(J,I))
                 H1(I) = H1(I+1)
 490           CONTINUE
               H1(I2+1) = 0.
 500       CONTINUE
C  ROTATION WITH THE ROWS N11+1,...N7
 510       DO 530 J=1,K1
             IF(WI.EQ.0.) GO TO 540
             IJ = N11+J
             IF(IJ.LE.0) GO TO 530
             PIV = H2(J)
C  CALCULATE THE PARAMETERS OF THE GIVENS TRANSFORMATION
             CALL COSSIN(PIV,WI,DPRIME(IJ),COS,SIN)
C  TRANSFORMATION TO THE RIGHT HAND SIDES.
             CALL ROTATE(PIV,COS,SIN,XI,CX(IJ))
             CALL ROTATE(PIV,COS,SIN,YI,CY(IJ))
             IF(J.EQ.K1) GO TO 540
             J1 = J+1
C  TRANSFORMATION TO THE LEFT HAND SIDE.
             DO 520 I=J1,K1
               CALL ROTATE(PIV,COS,SIN,H2(I),G2(IJ,I))
 520         CONTINUE
 530       CONTINUE
 540     CONTINUE
C  BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFF CX(J),J=1,...N7.
         CALL PBACK(G1,G2,CX,N7,K1,CX)
C  BACKWARD SUBSTITUTION TO OBTAIN THE B-SPLINE COEFF CY(J),J=1,...N7.
         CALL PBACK(G1,G2,CY,N7,K1,CY)
C  CALCULATE FROM CONDITION (**) THE COEFFICIENTS CX(N7+J) AND CY(N7+J),
C  J=1,2,...K.
         DO 545 I=1,K
           J = I+N7
           CX(J) = CX(I)
           CY(J) = CY(I)
 545     CONTINUE
C  COMPUTATION OF F(P).
         FP = 0.
         L = K1
         DO 570 IT=1,M1
           IF(Z(IT).LT.T(L)) GO TO 550
```

```
            L = L+1
  550       LO = L-K2
            TERM1 = 0.
            TERM2 = 0.
            DO 560 J=1,K1
               LC = LO+1
               TERM1 = TERM1+CX(LO)*Q(IT,J)
               TERM2 = TERM2+CY(LO)*Q(IT,J)
  560       CONTINUE
            FP = FP+W(IT)*((TERM1-X(IT))**2+(TERM2-Y(IT))**2)
  570     CONTINUE
C   TEST WHETHER THE APPROXIMATION X=SXP(Z),Y=SYP(Z) IS AN ACCEPTABLE
C   SOLUTION.
          FPMS = FP-S
          IF(ABS(FPMS).LT.ACC) GO TO 660
C   TEST WHETHER THE MAXIMAL NUMBER OF ITERATIONS IS REACHED.
          IF(ITER.EQ.MAXIT) GO TO 600
C   CARRY OUT ONE MORE STEP OF THE ITERATION PROCESS.
          P2 = P
          F2 = FPMS
          IF(ICHECK.NE.0) GO TO 580
          IF((F2-F3).GT.ACC) GO TO 575
C   OUR INITIAL CHOICE OF P IS TOO LARGE.
          P = P*0.1E-2
          P3 = P2
          F3 = F2
          GO TO 590
  575     IF((F1-F2).GT.ACC) GO TO 580
C   OUR INITIAL CHOICE OF P IS TOO SMALL
          P = P*0.1E+04
          P1 = P2
          F1 = F2
          GO TO 590
C   TEST WHETHER THE ITERATION PROCESS PROCEEDS AS THEORETICALLY
C   EXPECTED.
  580     IF(F2.GE.F1 .OR. F2.LE.F3) GO TO 610
          ICHECK = 1
C   FIND THE NEW VALUE FOR P.
          P = RATION(P1,F1,P2,F2,P3,F3)
  590   CONTINUE
C   ERROR CODES AND MESSAGES.
  600   IER = 3
        GO TO 660
  610   IER = 2
        GO TO 660
  620   IER = 1
        GO TO 660
  630   IER = -1
```

```
      GO TO 660
  640 IER = -2
C   THE APPROXIMATION X=C1,Y=C2, WITH C1 AND C2 THE LEAST-SQUARES
C   CONSTANT FUNCTIONS, IS A SOLUTION OF OUR PROBLEM.
C   A CONSTANT FUNCTION IS A SPLINE OF DEGREE K WITH ALL B-SPLINE
C   COEFFICIENTS EQUAL TO THAT CONSTANT.
      DO 650 I=1,K1
        T(I) = Z(1)
        CX(I) = C1
        CY(I) = C2
        J = I+K1
        T(J) = Z(M)
  650 CONTINUE
      N = NMIN
      FP = FPO
  660 RETURN
      END
      SUBROUTINE PBACK(A,B,Z,N,K,C)
C   SUBROUTINE PBACK CALCULATES THE SOLUTION OF THE SYSTEM OF EQUATIONS
C   G * C = Z  WITH G  A N X N UNIT UPPER TRIANGULAR MATRIX OF THE FORM
C                   A !
C          G =      ! B
C                   0 !
C   WITH B A N X K MATRIX AND A A (N-K) X (N-K) UNIT UPPER TRIANGULAR
C   MATRIX OF BANDWIDTH K+1.
C   ATTENTION: THE FIRST DIMENSION SPECIFICATION OF THE MATRICES
C   A AND B MUST BE THE SAME AS IN THE DRIVER PROGRAM.
      DIMENSION A(1000,K),B(1000,K),Z(N),C(N)
      N2 = N-K
      L = N
      DO 30 I=1,K
        STORE = Z(L)
        IF(I.EQ.1) GO TO 20
        J = K+2-I
        LO = L
        DO 10 L1=J,K
          LO = LO+1
          STORE = STORE-C(LO)*B(L,L1)
   10   CONTINUE
   20   C(L) = STORE
        L = L-1
        IF(L.EQ.0) GO TO 80
   30 CONTINUE
      DO 50 I=1,N2
        STORE = Z(I)
        L = N2
        DO 40 J=1,K
          L = L+1
```

```
          STORE = STORE-C(L)*B(I,J)
40      CONTINUE
        C(I) = STORE
50    CONTINUE
      I = N2
      IF(I.EQ.1) GO TO 80
      DO 70 J=2,N2
        I = I-1
        STORE = C(I)
        I1 = MIN0(K,J-1)
        L = I
        DO 60 LO=1,I1
          L = L+1
          STORE = STORE-C(L)*A(I,LO)
60      CONTINUE
        C(I) = STORE
70    CONTINUE
80    RETURN
      END
```

```
C PROGRAM FILTREX.FOR
C
C OWNER CESAR TADEU DE MIRANDA
C DATE 06/OCT/83
C
C THIS IS A FORTRAN PROGRAM THAT FILTERS THE FILE OF SELECTED KNOTS
C FOR022.DAT PRODUCED BY THE DIERKS IMPLEMENTATION PROGRAM TH8KNT.FOR.
C THE FILTERING IS DONE IN THE FOLLOWING SENSE: THE KNOTS WHICH ARE
C REPEATED DUE TO THE ROUND OFF ARE TRANSFORMED INTO MULTIPLICTY ONE
C KNOTS BY PRUNNING THE REPEATED OCCURENCIES.
C
C INPUT DATA FROM FILE FOR022.DAT IN FORMAT(2I5)
C OUPUT DATA INTO FILE FOR023.DAT IN FORMAT(2I5)
C
C VARIABLES DECLARATION
      INTEGER TEMPX(1000),TEMPY(1000),TOTKNT,NSUM,IR,IW,CONT
      INTEGER NEWI,I,N,ITYPE
C     INTITIALIZATION
      DATA IR,IW/22,23/
      NSUM=0
      CONT=0
      TOTKNT=0
      READ(IR,124) IX,IY
      DO WHILE (IX.NE.-3)
        N=IX
        NSUM=NSUM+N
        ITYPE=IY
        CONT=CONT+1
        READ(IR,124)IX,IY
        TYPE *,' '
        TYPE *,'CONTOUR #',CONT
        TYPE *,'INITIAL # OF POINTS',N,'  TYPE',ITYPE
        TEMPX(1)=IX
        TEMPY(1)=IY
        NEWI=1
        DO I=2,N
          READ(IR,124) IX,IY
          IF((TEMPX(NEWI).NE.IX).AND.(TEMPY(NEWI).NE.IY)) THEN
            NEWI=NEWI+1
            TEMPX(NEWI)=IX
            TEMPY(NEWI)=IY
          END IF
        END DO
        TOTKNT=TOTKNT+NEWI
        WRITE(IW,124) NEWI,ITYPE
        TYPE *,'REDUCED # OF KNOTS',NEWI
        DO I=1,NEWI
          WRITE(IW,124) TEMPX(I),TEMPY(I)
```

```
          END CC
          READ(IR,124)IX,IY
        END DC
C ADC SENTINNEL
        IX=-3
        IY=-3
        WRITE(IW,124)IX,IY
        TYPE *,'TOTAL NUMBER OF INITIAL KNOTS',NSUM
        TYPE *,'TOTAL NUMBER OF REDUCED KNOTS',TOTKNT
        TYPE *,'END OF PROGRAM'
        STOP
 124    FORMAT(2I5)
        END
```

```
C PROGRAM FRASM.FOR
C
C OWNER CESAR TADEU DE MIRANDA
C DATE 04/OCT/83
C
C THE OBJECTIVE OF THIS FORTRAN PROGRAM IS TO APPLY THE FRANKE
C TRIANGULATION ALGORITHM FOR INTERPOLATION OF SCATTERED DATA.
C
C INPUT PARAMETERS
C     FILE FOR019.DAT: FILE CONTAINING THE NUMBERS OF CONTOURS PER EACH
C     LEVEL AND THE GRAY LEVEL VALUE OF EACH LEVEL.
C     FILE FOR023.DAT: FILE CONTAINING THE SELECTED KONTS REPRESENTING
C     THE IMAGE AS A SCATTERED SET OF DATA.
C
C OUTPUT PARAMETERS
C     FILE FOR031.DAT: UNIFORM GRID (256,256) REPRESENTING THE RECONS-
C     TRUCTED IMAGE.
C
C VARIABLE DECLARATION
      REAL X(15000),Y(15000),F(15000),XO(256),YO(256),FO(256,256)
      REAL WK(130000),CLEV
      INTEGER L,K,NCONT,I,N,IX,IY,ITYPE,IER,TCONT,IWK(480000)
      INTEGER NXO,NYO,J,IMIN,IMAX,ITEMP
      INTEGER MX(15000),MY(15000),KT,FLAG,I1,IPROV(256),IST,IEND
      REAL GI,GA
C
C ERROR CODE
C     ERROR: 0 NORMAL TERMINATION
C           -1 DIMENSION IS NOT ENOUGH
C            POSITIVE VALUES OF THE ERROR ARE GENERATED BY THE
C            SUBROUTINE QSHEP, SEE DESCRIPTION OF THESE ERRORS
C            IMBED IN THAT BLOCK.
C ENTER THE DATA
      L=0
      K=1
      IER=0
      TCONT=0
      TYPE *,'ENTER UNIT NUMBER OF CONTOURS IR'
C      ACCEPT *,IR(TO RUN ON BATCH)
      IR=23
      READ(19,100) NCONT,CLEV
      GI=CLEV
      DO WHILE ((NCONT.NE.-3).AND.(IER.EQ.0))
        TYPE *,'LEVEL # ',K
        TYPE *,' '
        TCONT=TCONT+NCONT
        IF(NCONT.NE.0) THEN
          TYPE *,'# OF CONTOURS IN THIS LEVEL:',NCONT,'  VALUE',CLEV
```

```
            DO I=1,NCONT
              READ(IR,124) IX,IY
C       TYPE *,'IX',IX,'    IY',IY
              N=IX
              ITYPE=IY
              DO J=1,N
                READ(IR,124)IX,IY
                L=L+1
                IF(L.GT.15000) THEN
                  IER=-1
                  TYPE *,' ERROR    ',IER
                  TYPE *,' # OF CONTOURS OF INPUT EXCEEDS THE DIMENSION
     1            OF ARRAYS MX, MY,F'
                  STOP
                END IF
                MX(L)=IX
                MY(L)=IY
                F(L)=CLEV
              END DO
            END DO
          ELSE
            TYPE *,'THE LEVEL',K,' HAS NO CONTOURS'
          END IF
          K=K+1
          GA=CLEV
          READ(15,100)NCONT,CLEV
        END DO
C
C ELIMINATE THE REPEATED ORDINATE PAIRS (MX,MY)
        X(1)=FLOAT(MX(1))
        Y(1)=FLOAT(MY(1))
        KT=1
        DO I=2,L
          FLAG=0
          I1=I-1
          DO J=I1,1,-1
            IF((MX(I).EQ.MX(J)).AND.(MY(I).EQ.MY(J))) THEN
              FLAG=1
C       TYPE *,'TRAP A ALLIEN'
              GO TO 20
            END IF
          END DO
 20       CONTINUE
          IF(FLAG.EQ.0) THEN
            KT=KT+1
            X(KT)=FLOAT(MX(I))
            Y(KT)=FLOAT(MY(I))
            F(KT)=F(I)
```

```
          END IF
        END DO
        TYPE *,'TOTAL NUMBER OF LEVELS',K-1
        TYPE *,'TOTAL NUMBER OF CONTOURS',TCONT
        TYPE *,'TOTAL NUMBER OF POINTS',L
        TYPE *,'TOTAL NUMBER OF POINTS WITHOUT REPETITION',KT
        TYPE *,'LOWER BOUND',GI,'  UPPER BOUND',GA
C
C ADDING THE FRANKE PARAMETERS
        IF(KT.GT.15000) THEN
          TYPE *,'THE NUMBER OF POINTS EXCEED 15000 '
          STOP
        END IF
        NPI=KT
        NXO=256
        NYO=256
        DO I=1,256
          XO(I)=FLOAT(I)
          YO(I)=XO(I)
        END DO
        TYPE *,'INVOKE FRANKE ALGORITHNM'
        CALL PWLOT(X,Y,F,NPI,XO,NXO,YO,NYO,FO,IWK,WK)
        TYPE *,'FINISH FRANKE ALGORITHM'
        TYPE *,'ERROR',IER
C
C FIND MAX AND MIN OF FO
        IMIN=NINT(FO(1,1))
        IMAX=NINT(FO(1,1))
        DO I=1,256
          DO J=1,256
            ITEMP=NINT(FO(I,J))
            IF(ITEMP.GT.IMAX) THEN
              IMAX=ITEMP
            ELSE IF (ITEMP.LT.IMIN) THEN
              IMIN=ITEMP
            END IF
          END DO
        END DO
        TYPE *,'IMAX',IMAX
        TYPE *,'IMIN',IMIN
        IRANGE=IMAX-IMIN
        TYPE *,'RANGE OF GRAY LEVEL',IRANGE
C
C NORMALIZATION
        TYPE *,'START THE NORMALIZATION'
        DO I=1,256
          DO J=1,256
            IF(FO(I,J).LT.GI) THEN
```

```
                FO(I,J)=GI
              ELSE IF(FO(I,J).GT.GA) THEN
                FO(I,J)=GA
              END IF
            END DO
          END DO
          TYPE *,'END NORMALIZATION'
C
C STORE THE MATRIX IN A FILE 31
          DO I=1,256
            WRITE(31,130) I
            DO J=1,256
              IPROV(J)=NINT(FO(I,J))
            END DO
            DO K=0,7
              IST=32*K +1
              IEND=IST+31
              WRITE(31,128)(IPROV(J),J=IST,IEND)
            END DO
          END DO
          TYPE *,'END OF PROGRAM'
          STOP
100       FORMAT(I5,F10.5)
124       FORMAT(2I5)
128       FORMAT(32I4)
130       FORMAT(I4)
          END
```

```
      SUBROUTINE PWLOT(X,Y,FI,NPI,XO,NXO,YO,NYO,FO,IWK,WK)
C
C     THIS SUBROUTINE CONSTRUCTS A PIECEWISE LINEAR SURFACE OVER A
C     TRIANGULATION OF THE SET OF DATA POINTS.  EXTRAPOLATION IS
C     PROVIDED FOR BY SETTING THE VALUE TO THE VALUE AT THE NEAREST
C     POINT IN THE CONVEX HULL.  AT LEAST THIS IS STABLE.
C
C     THIS PROGRAM WAS WRITTEN BY RICHARD FRANKE
C                                 408/646-2758
C                            DEPARTMENT OF MATHEMATICS
C                            NAVAL POSTGRADUATE SCHOOL
C                            MONTEREY, CA 93943
C
C     OCTOBER 1983
C
C     THE PARAMETERS IN THE CALLING SEQUENCE ARE AS FOLLOWS
C
C     INPUT PARAMETERS
C
C        X,Y,FI    - THE NPI DATA POINTS
C        NPI       - THE NUMBER OF DATA POINTS
C        XO        - AN ARRAY OF X VALUES FOR THE OUTPUT GRID
C        NXO       - THE NUMBER OF VALUES IN XO
C        YO        - AN ARRAY OF Y VALUES FOR THE OUTPUT GRID
C        NYO       - THE NUMBER OF VALUES IN YO
C
C     OUTPUT PARAMETERS
C
C        FO        - AN NXO BY NYO GRID OF FUNCTION VALUES.  THE ARRAY
C                    IS ASSUMED TO BE DIMENSION (NXO,NYO) IN THE
C                    CALLING PROGRAM
C
C     WORKSPACE ARRAYS
C
C        IWK       - MUST BE DIMENSIONED AT LEAST 31*NPI IN THE CALLING
C                    PROGRAM
C        WK        - MUST BE DIMENSIONED AT LEAST 8*NPI IN THE CALLING
C                    PROGRAM
C
      DIMENSION X(1),Y(1),FI(1),XO(1),YO(1),FO(NXO,NYO),WK(1),IWK(1)
      COMMON/IDLC/NIT
      NIT = 0
C     TRIANGULATE THE DATA SET
      CALL IDTANG(NPI,X,Y,NT,IWK,NL,IWK(6*NPI+1),IWK(12*NPI+1),
     1 IWK(30*NPI+1),WK)
      TYPE *,'NUMBER OF TRIANGLES',NT
      TYPE *,'NUMBER OF BORDER SEG',NL
```

```
C       EVALUATE ON THE GRID
        DO 400 J=1,NYO
          DO 300 I=1,NXO
C           GET THE TRIANGLE NUMBER, OR BORDER SEGMENT(S)
            CALL IDLCTN(NPI,X,Y,NT,IWK,NL,IWK(6*NPI+1),XO(I),YO(J),ITI,
     1       IWK(12*NPI+1),WK)
C               CHECK:  IN A TRIANGLE, OR OUTSIDE CONVEX HULL?
            IF(ITI.GT.NT)GO TO 200
C               IN TRIANGLE NT
              CALL PWLIN(X,Y,FI,IWK,ITI,XO(I),YO(J),FV)
              GO TO 250
  200       CONTINUE
C               OUTSIDE CONVEX HULL, EXTRAPOLATE
            CALL EXTR(X,Y,FI,NPI,NT,NL,ITI,IWK(6*NPI+1),XO(I),YO(J),FV)
  250       FO(I,J) = FV
  300     CONTINUE
  400   CONTINUE
        RETURN
        END
        SUBROUTINE EXTR(X,Y,FI,NPI,NT,NL,ITI,IPL,XV,YV,FV)
C       EXTRAPOLATE OUTSIDE CONVEX HULL OF DATA SET
        DIMENSION X(1),Y(1),FI(1),IPL(1)
        FV = 0.
        NTL = NT + NL
        IL1 = ITI/NTL
        IF(IL1.LE.0)RETURN
        IL2 = ITI - IL1*NTL
        JIPL = 3*IL2 - 2
        I = IPL(JIPL)
        IF(IL1.EQ.IL2)GO TO 200
        FV = FI(I)
        RETURN
  200   J = IPL(JIPL+1)
        XD = X(J) - X(I)
        YD = Y(J) - Y(I)
        E = SQRT(XD*XD + YD*YD)
        EP = ((XV - X(I))*XD + (YV - Y(I))*YD)/E
        P = EP/E
        FV = P*FI(J) + (1. - P)*FI(I)
        RETURN
        END
        SUBROUTINE PWLIN(X,Y,FI,IPT,ITI,XV,YV,FV)
C       INTERPOLATE USING LINEAR INTERPOLATION WITHIN THE TRIANGLE
        DIMENSION X(1),Y(1),FI(1),B(3),IPT(3,1),FN(6),XC(3),YC(3)
        ACF(X1,Y1,X2,Y2,X3,Y3) = (X1 - X2)*(Y1 - Y3) - (X1 - X3)*(Y1 - Y2)
        IND(I) = MOD(I-1,3) + 1
        DO 100 L=1,3
        I = IPT(L,ITI)
```

```
      FN(L) = FI(I)
      XC(L) = X(I)
  100 YC(L) = Y(I)
      DEL = ACF(XC(1),YC(1),XC(2),YC(2),XC(3),YC(3))
      FV = 0.
      DO 110 I=1,3
      J = IND(I+1)
      K = IND(I+2)
      B(I) = ACF(XV,YV,XC(J),YC(J),XC(K),YC(K))/DEL
  110 FV = FV + B(I)*FN(I)
      RETURN
      END
      SUBROUTINE  IDLCTN(NDP,XD,YD,NT,IPT,NL,IPL,XII,YII,ITI,
     1                   IWK,WK)
C THIS SUBROUTINE LOCATES A POINT, I.E., DETERMINES TO WHAT TRI-
C ANGLE A GIVEN POINT (XII,YII) BELONGS.  WHEN THE GIVEN POINT
C DOES NOT LIE INSIDE THE DATA AREA, THIS SUBROUTINE DETERMINES
C THE BORDER LINE SEGMENT WHEN THE POINT LIES IN AN OUTSIDE
C RECTANGULAR AREA, AND TWO BORDER LINE SEGMENTS WHEN THE POINT
C LIES IN AN OUTSIDE TRIANGULAR AREA.
C THE INPUT PARAMETERS ARE
C      NDP =  NUMBER OF DATA POINTS,
C      XD,YD  = ARRAYS OF DIMENSION NDP CONTAINING THE X AND Y
C               COORDINATES OF THE DATA POINTS,
C      NT   =  NUMBER OF TRIANGLES,
C      IPT =  INTEGER ARRAY OF DIMENSION 3*NT CONTAINING THE
C               POINT NUMBERS OF THE VERTEXES OF THE TRIANGLES,
C      NL   =  NUMBER OF BORDER LINE SEGMENTS,
C      IPL =  INTEGER ARRAY OF DIMENSION 3*NL CONTAINING THE
C               POINT NUMBERS OF THE END POINTS OF THE BORDER
C               LINE SEGMENTS AND THEIR RESPECTIVE TRIANGLE
C               NUMBERS,
C      XII,YII = X AND Y COORDINATES OF THE POINT TO BE
C               LOCATED.
C THE OUTPUT PARAMETER IS
C      ITI =  TRIANGLE NUMBER, WHEN THE POINT IS INSIDE THE
C               DATA AREA, OR
C               TWO BORDER LINE SEGMENT NUMBERS, IL1 AND IL2,
C               CODED TO IL1*(NT+NL)+IL2, WHEN THE POINT IS
C               OUTSIDE THE DATA AREA.
C THE OTHER PARAMETERS ARE
C      IWK =  INTEGER ARRAY OF DIMENSION 18*NDP USED INTER-
C               NALLY AS A WORK AREA,
C      WK   =  ARRAY OF DIMENSION 8*NDP USED INTERNALLY AS A
C               WORK AREA.
C DECLARATION STATEMENTS
      DIMENSION   XD(15000),YD(15000),IPT(40000),IPL(10000),
     1            IWK(270000),WK(130000)
```

```
      DIMENSION   NTSC(9),IDSC(9)
      COMMON/IDLC/NIT
C STATEMENT FUNCTIONS
      SIDE(U1,V1,U2,V2,U3,V3)=(U1-U3)*(V2-V3)-(V1-V3)*(U2-U3)
      SPDT(U1,V1,U2,V2,U3,V3)=(U1-U2)*(U3-U2)+(V1-V2)*(V3-V2)
C PRELIMINARY PROCESSING
   10 NDPO=NDP
      NTO=NT
      NLO=NL
      NTL=NTO+NLO
      XO=XII
      YO=YII
C PROCESSING FOR A NEW SET OF DATA POINTS
   20 IF(NIT.NE.0)   GO TO 30
      NIT=1
C - DIVIDES THE X-Y PLANE INTO NINE RECTANGULAR SECTIONS.
      XMN=XD(1)
      XMX=XMN
      YMN=YD(1)
      YMX=YMN
      DO 21   IDP=2,NDPO
        XI=XD(IDP)
        YI=YD(IDP)
        XMN=AMIN1(XI,XMN)
        XMX=AMAX1(XI,XMX)
        YMN=AMIN1(YI,YMN)
        YMX=AMAX1(YI,YMX)
   21 CONTINUE
      XS1=(XMN+XMN+XMX)/3.0
      XS2=(XMN+XMX+XMX)/3.0
      YS1=(YMN+YMN+YMX)/3.0
      YS2=(YMN+YMX+YMX)/3.0
C - DETERMINES AND STORES IN THE IWK ARRAY TRIANGLE NUMBERS OF
C - THE TRIANGLES ASSOCIATED WITH EACH OF THE NINE SECTIONS.
      DO 22   ISC=1,9
        NTSC(ISC)=0
        IDSC(ISC)=0
   22 CONTINUE
      ITOT3=0
      JWK=0
      DO 27   ITO=1,NTO
        ITOT3=ITOT3+3
        I1=IPT(ITOT3-2)
        I2=IPT(ITOT3-1)
        I3=IPT(ITOT3)
        XMN=AMIN1(XD(I1),XD(I2),XD(I3))
        XMX=AMAX1(XD(I1),XD(I2),XD(I3))
        YMN=AMIN1(YD(I1),YD(I2),YD(I3))
```

```
      YMX=AMAX1(YD(I1),YD(I2),YD(I3))
      IF(YMN.GT.YS1)                          GO TO 23
      IF(XMN.LE.XS1)                          IDSC(1)=1
      IF(XMX.GE.XS1.AND.XMN.LE.XS2)           IDSC(2)=1
      IF(XMX.GE.XS2)                          IDSC(3)=1
   23 IF(YMX.LT.YS1.OR.YMN.GT.YS2)            GO TO 24
      IF(XMN.LE.XS1)                          IDSC(4)=1
      IF(XMX.GE.XS1.AND.XMN.LE.XS2)           IDSC(5)=1
      IF(XMX.GE.XS2)                          IDSC(6)=1
   24 IF(YMX.LT.YS2)                          GO TO 25
      IF(XMN.LE.XS1)                          IDSC(7)=1
      IF(XMX.GE.XS1.AND.XMN.LE.XS2)           IDSC(8)=1
      IF(XMX.GE.XS2)                          IDSC(9)=1
   25 DO 26  ISC=1,9
        IF(IDSC(ISC).EQ.0)   GO TO 26
        JIWK=9*NTSC(ISC)+ISC
        IWK(JIWK)=ITO
        NTSC(ISC)=NTSC(ISC)+1
        IDSC(ISC)=0
   26 CONTINUE
C - STORES IN THE WK ARRAY THE MINIMUM AND MAXIMUM OF THE X AND
C - Y COORDINATE VALUES FOR EACH OF THE TRIANGLE.
      JWK=JWK+4
      WK(JWK-3)=XMN
      WK(JWK-2)=XMX
      WK(JWK-1)=YMN
      WK(JWK)  =YMX
   27 CONTINUE
      GO TO 60
C CHECKS IF IN THE SAME TRIANGLE AS PREVIOUS.
   30 ITO=ITIPV
      IF(ITO.GT.NTO)       GO TO 40
      ITOT3=ITO*3
      IP1=IPT(ITOT3-2)
      X1=XD(IP1)
      Y1=YD(IP1)
      IP2=IPT(ITOT3-1)
      X2=XD(IP2)
      Y2=YD(IP2)
      IF(SIDE(X1,Y1,X2,Y2,XO,YO).LT.0.0)       GO TO 60
      IP3=IPT(ITOT3)
      X3=XD(IP3)
      Y3=YD(IP3)
      IF(SIDE(X2,Y2,X3,Y3,XO,YO).LT.0.0)       GO TO 60
      IF(SIDE(X3,Y3,X1,Y1,XO,YO).LT.0.0)       GO TO 60
      GO TO 80
C CHECKS IF ON THE SAME BORDER LINE SEGMENT.
   40 IL1=ITO/NTL
```

```
      IL2=ITO-IL1*NTL
      IL1T3=IL1*3
      IP1=IPL(IL1T3-2)
      X1=XD(IP1)
      Y1=YD(IP1)
      IP2=IPL(IL1T3-1)
      X2=XD(IP2)
      Y2=YD(IP2)
      IF(IL2.NE.IL1)          GO TO 50
      IF(SPDT(X1,Y1,X2,Y2,XO,YO).LT.0.0)        GO TO 60
      IF(SPDT(X2,Y2,X1,Y1,XO,YO).LT.0.0)        GO TO 60
      IF(SIDE(X1,Y1,X2,Y2,XO,YO).GT.0.0)        GO TO 60
      GO TO 80
C CHECKS IF BETWEEN THE SAME TWO BORDER LINE SEGMENTS.
   50 IF(SPDT(X1,Y1,X2,Y2,XO,YO).GT.0.0)        GO TO 60
      IP3=IPL(3*IL2-1)
      X3=XD(IP3)
      Y3=YD(IP3)
      IF(SPDT(X3,Y3,X2,Y2,XO,YO).LE.0.0)        GO TO 80
C LOCATES INSIDE THE DATA AREA.
C - DETERMINES THE SECTION IN WHICH THE POINT IN QUESTION LIES.
   60 ISC=1
      IF(XO.GE.XS1)           ISC=ISC+1
      IF(XO.GE.XS2)           ISC=ISC+1
      IF(YO.GE.YS1)           ISC=ISC+3
      IF(YO.GE.YS2)           ISC=ISC+3
C - SEARCHES THROUGH THE TRIANGLES ASSOCIATED WITH THE SECTION.
      NTSCI=NTSC(ISC)
      IF(NTSCI.LE.0)          GO TO 70
      JIWK=-9*ISC
      DO 61   ITSC=1,NTSCI
        JIWK=JIWK+9
        ITO=IWK(JIWK)
        JWK=ITO*4
        IF(XO.LT.WK(JWK-3))     GO TO 61
        IF(XO.GT.WK(JWK-2))     GO TO 61
        IF(YO.LT.WK(JWK-1))     GO TO 61
        IF(YO.GT.WK(JWK))       GO TO 61
        ITOT3=ITO*3
        IP1=IPT(ITOT3-2)
        X1=XD(IP1)
        Y1=YD(IP1)
        IP2=IPT(ITOT3-1)
        X2=XD(IP2)
        Y2=YD(IP2)
        IF(SIDE(X1,Y1,X2,Y2,XO,YO).LT.0.0)    GO TO 61
        IP3=IPT(ITOT3)
        X3=XD(IP3)
```

```
            Y3=YC(IP3)
            IF(SIDE(X2,Y2,X3,Y3,X0,Y0).LT.0.0)     GO TO 61
            IF(SIDE(X3,Y3,X1,Y1,X0,Y0).LT.0.0)     GO TO 61
            GO TO 80
   61 CONTINUE
C  LOCATES OUTSIDE THE DATA AREA.
   70 DO 72   IL1=1,NLO
            IL1T3=IL1*3
            IP1=IPL(IL1T3-2)
            X1=XD(IP1)
            Y1=YD(IP1)
            IP2=IPL(IL1T3-1)
            X2=XD(IP2)
            Y2=YD(IP2)
            IF(SPDT(X2,Y2,X1,Y1,X0,Y0).LT.0.0)     GO TO 72
            IF(SPDT(X1,Y1,X2,Y2,X0,Y0).LT.0.0)     GO TO 71
            IF(SIDE(X1,Y1,X2,Y2,X0,Y0).GT.0.0)     GO TO 72
            IL2=IL1
            GO TO 75
   71       IL2=MCD(IL1,NLO)+1
            IP3=IPL(3*IL2-1)
            X3=XC(IP3)
            Y3=YC(IP3)
            IF(SPDT(X3,Y3,X2,Y2,X0,Y0).LE.0.0)     GO TO 75
   72 CONTINUE
            ITO=1
            GO TO 80
   75 ITO=IL1*NTL+IL2
C  NORMAL EXIT
   80 ITI=ITO
            ITIPV=ITO
            RETURN
            END
            SUBROUTINE  IDTANG(NDP,XD,YD,NT,IPT,NL,IPL,IWL,IWP,WK)
C  THIS SUBROUTINE PERFORMS TRIANGULATION.  IT DIVIDES THE X-Y
C  PLANE INTO A NUMBER OF TRIANGLES ACCORDING TO GIVEN DATA
C  POINTS IN THE PLANE, DETERMINES LINE SEGMENTS THAT FORM THE
C  BORDER OF DATA AREA, AND DETERMINES THE TRIANGLE NUMBERS
C  CORRESPONDING TO THE BORDER LINE SEGMENTS.
C  AT COMPLETION, POINT NUMBERS OF THE VERTEXES OF EACH TRIANGLE
C  ARE LISTED COUNTER-CLOCKWISE.  POINT NUMBERS OF THE END POINTS
C  OF EACH BORDER LINE SEGMENT ARE LISTED COUNTER-CLOCKWISE,
C  LISTING ORDER OF THE LINE SEGMENTS BEING COUNTER-CLOCKWISE.
C  THE LUN CONSTANT IN THE DATA INITIALIZATION STATEMENT IS THE
C  LOGICAL UNIT NUMBER OF THE STANDARD OUTPUT UNIT AND IS,
C  THEREFORE, SYSTEM DEPENDENT.
C  THIS SUBROUTINE CALLS THE IDXCHG FUNCTION.
C  THE INPUT PARAMETERS ARE
```

```
C       NDP = NUMBER OF DATA POINTS,
C       XD  = ARRAY OF DIMENSION NDP CONTAINING THE
C             X COORDINATES OF THE DATA POINTS,
C       YD  = ARRAY OF DIMENSION NDP CONTAINING THE
C             Y COORDINATES OF THE DATA POINTS.
C THE OUTPUT PARAMETERS ARE
C       NT  = NUMBER OF TRIANGLES,
C       IPT = INTEGER ARRAY OF DIMENSION 6*NDP-15, WHERE THE
C             POINT NUMBERS OF THE VERTEXES OF THE (IT)TH
C             TRIANGLE ARE TO BE STORED AS THE (3*IT-2)ND,
C             (3*IT-1)ST, AND (3*IT)TH ELEMENTS,
C             IT=1,2,...,NT,
C       NL  = NUMBER OF BORDER LINE SEGMENTS,
C       IPL = INTEGER ARRAY OF DIMENSION 6*NDP, WHERE THE
C             POINT NUMBERS OF THE END POINTS OF THE (IL)TH
C             BORDER LINE SEGMENT AND ITS RESPECTIVE TRIANGLE
C             NUMBER ARE TO BE STORED AS THE (3*IL-2)ND,
C             (3*IL-1)ST, AND (3*IL)TH ELEMENTS,
C             IL=1,2,...,NL.
C THE OTHER PARAMETERS ARE
C       IWL = INTEGER ARRAY OF DIMENSION 18*NDP USED
C             INTERNALLY AS A WORK AREA,
C       IWP = INTEGER ARRAY OF DIMENSION NDP USED
C             INTERNALLY AS A WORK AREA,
C       WK  = ARRAY OF DIMENSION NDP USED INTERNALLY AS A
C             WORK AREA.
C DECLARATION STATEMENTS
      DIMENSION   XD(15000),YD(15000),IPT(40000),IPL(1000),
     1            IWL(270000),IWP(15000),WK(15000)
      DIMENSION   ITF(2)
      DATA  RATIO/1.0E-6/, NREP/100/, LUN/6/
C STATEMENT FUNCTIONS
      DSQF(U1,V1,U2,V2)=(U2-U1)**2+(V2-V1)**2
      SIDE(U1,V1,U2,V2,U3,V3)=(V3-V1)*(U2-U1)-(U3-U1)*(V2-V1)
C PRELIMINARY PROCESSING
   10 NDP0=NDP
      NDPM1=NDP0-1
      IF(NDP0.LT.4)         GO TO 90
C DETERMINES THE CLOSEST PAIR OF DATA POINTS AND THEIR MIDPOINT.
   20 DSQMN=DSQF(XD(1),YD(1),XD(2),YD(2))
      IPMN1=1
      IPMN2=2
      DO 22  IP1=1,NDPM1
        X1=XD(IP1)
        Y1=YD(IP1)
        IP1P1=IP1+1
        DO 21  IP2=IP1P1,NDP0
          DSQI=DSQF(X1,Y1,XD(IP2),YD(IP2))
```

```
            IF(DSQI.EQ.0.0)       GO TO 91
            IF(DSQI.GE.DSQMN)     GO TO 21
            DSQMN=DSQI
            IPMN1=IP1
            IPMN2=IP2
   21     CONTINUE
   22   CONTINUE
        DSQ12=DSQMN
        XDMP=(XD(IPMN1)+XD(IPMN2))/2.0
        YDMP=(YD(IPMN1)+YD(IPMN2))/2.0
C  SORTS THE OTHER (NDP-2) DATA POINTS IN ASCENDING ORDER OF
C  DISTANCE FROM THE MIDPOINT AND STORES THE SORTED DATA POINT
C  NUMBERS IN THE IWP ARRAY.
   30   JP1=2
        DO 31   IP1=1,NDPO
          IF(IP1.EQ.IPMN1.OR.IP1.EQ.IPMN2)       GO TO 31
          JP1=JP1+1
          IWP(JP1)=IP1
          WK(JP1)=DSQF(XDMP,YDMP,XD(IP1),YD(IP1))
   31   CONTINUE
        DO 33   JP1=3,NDPM1
          DSQMN=WK(JP1)
          JPMN=JP1
          DO 32   JP2=JP1,NDPO
            IF(WK(JP2).GE.DSQMN)       GO TO 32
            DSQMN=WK(JP2)
            JPMN=JP2
   32     CONTINUE
          ITS=IWP(JP1)
          IWP(JP1)=IWP(JPMN)
          IWP(JPMN)=ITS
          WK(JPMN)=WK(JP1)
   33   CONTINUE
C  IF NECESSARY, MODIFIES THE ORDERING IN SUCH A WAY THAT THE
C  FIRST THREE DATA POINTS ARE NOT COLLINEAR.
   35   AR=DSQ12*RATIO
        X1=XD(IPMN1)
        Y1=YD(IPMN1)
        DX21=XD(IPMN2)-X1
        DY21=YD(IPMN2)-Y1
        DO 36   JP=3,NDPO
          IP=IWP(JP)
          IF(ABS((YD(IP)-Y1)*DX21-(XD(IP)-X1)*DY21).GT.AR)
     1                    GO TO 37
   36   CONTINUE
        GO TO 52
   37   IF(JP.EQ.3)     GO TO 40
        JPMX=JP
```

```
      JP=JPMX+1
      DO 38   JPC=4,JPMX
        JP=JP-1
        IWP(JP)=IWP(JP-1)
   38 CONTINUE
      IWP(3)=IP
C FORMS THE FIRST TRIANGLE.  STORES POINT NUMBERS OF THE VER-
C TEXES OF THE TRIANGLE IN THE IPT ARRAY, AND STORES POINT NUM-
C BERS OF THE BORDER LINE SEGMENTS AND THE TRIANGLE NUMBER IN
C THE IPL ARRAY.
   40 IP1=IPMN1
      IP2=IPMN2
      IP3=IWP(3)
      IF(SIDE(XD(IP1),YD(IP1),XD(IP2),YD(IP2),XD(IP3),YD(IP3))
     1     .GE.0.0)        GO TO 41
      IP1=IPMN2
      IP2=IPMN1
   41 NTO=1
      NTT3=3
      IPT(1)=IP1
      IPT(2)=IP2
      IPT(3)=IP3
      NLO=3
      NLT3=9
      IPL(1)=IP1
      IPL(2)=IP2
      IPL(3)=1
      IPL(4)=IP2
      IPL(5)=IP3
      IPL(6)=1
      IPL(7)=IP3
      IPL(8)=IP1
      IPL(9)=1
C ADDS THE REMAINING (NDP-3) DATA POINTS, ONE BY ONE.
   50 DO 79   JP1=4,NDPO
        IP1=IWP(JP1)
        X1=XD(IP1)
        Y1=YD(IP1)
C - DETERMINES THE VISIBLE BORDER LINE SEGMENTS.
        IP2=IPL(1)
        JPMN=1
        DXMN=XD(IP2)-X1
        DYMN=YD(IP2)-Y1
        DSQMN=DXMN**2+DYMN**2
        ARMN=DSQMN*RATIO
        JPMX=1
        DXMX=DXMN
        DYMX=DYMN
```

```
            DSQMX=DSQMN
            ARMX=ARMN
            DO 52  JP2=2,NLO
              IP2=IPL(3*JP2-2)
              DX=XC(IP2)-X1
              DY=YC(IP2)-Y1
              AR=DY*DXMN-DX*DYMN
              IF(AR.GT.ARMN)        GO TO 51
              DSQI=DX**2+DY**2
              IF(AR.GE.(-ARMN).AND.DSQI.GE.DSQMN)       GO TO 51
              JPMN=JP2
              DXMN=DX
              DYMN=DY
              DSQMN=DSQI
              ARMN=DSQMN*RATIO
   51         AR=DY*DXMX-DX*DYMX
              IF(AR.LT.(-ARMX))     GO TO 52
              DSQI=DX**2+DY**2
              IF(AR.LE.ARMX.AND.DSQI.GE.DSQMX)     GO TO 52
              JPMX=JP2
              DXMX=DX
              DYMX=DY
              DSQMX=DSQI
              ARMX=DSQMX*RATIO
   52       CONTINUE
            IF(JPMX.LT.JPMN)   JPMX=JPMX+NLO
            NSH=JPMN-1
            IF(NSH.LE.0)        GO TO 60
C - SHIFTS (ROTATES) THE IPL ARRAY TO HAVE THE INVISIBLE BORDER
C - LINE SEGMENTS CONTAINED IN THE FIRST PART OF THE IPL ARRAY.
            NSHT3=NSH*3
            DO 53  JP2T3=3,NSHT3,3
              JP3T3=JP2T3+NLT3
              IPL(JP3T3-2)=IPL(JP2T3-2)
              IPL(JP3T3-1)=IPL(JP2T3-1)
              IPL(JP3T3)  =IPL(JP2T3)
   53       CONTINUE
            DO 54  JP2T3=3,NLT3,3
              JP3T3=JP2T3+NSHT3
              IPL(JP2T3-2)=IPL(JP3T3-2)
              IPL(JP2T3-1)=IPL(JP3T3-1)
              IPL(JP2T3)  =IPL(JP3T3)
   54       CONTINUE
            JPMX=JPMX-NSH
C - ADDS TRIANGLES TO THE IPT ARRAY, UPDATES BORDER LINE
C - SEGMENTS IN THE IPL ARRAY, AND SETS FLAGS FOR THE BORDER
C - LINE SEGMENTS TO BE REEXAMINED IN THE IWL ARRAY.
   60       JWL=0
```

```
          DO 64  JP2=JPMX,NLO
            JP2T3=JP2*3
            IPL1=IPL(JP2T3-2)
            IPL2=IPL(JP2T3-1)
            IT  =IPL(JP2T3)
C - - ADDS A TRIANGLE TO THE IPT ARRAY.
            NTO=NTO+1
            NTT3=NTT3+3
            IPT(NTT3-2)=IPL2
            IPT(NTT3-1)=IPL1
            IPT(NTT3)  =IP1
C - - UPDATES BORDER LINE SEGMENTS IN THE IPL ARRAY.
            IF(JP2.NE.JPMX)       GO TO 61
            IPL(JP2T3-1)=IP1
            IPL(JP2T3)  =NTO
   61       IF(JP2.NE.NLO)        GO TO 62
            NLN=JPMX+1
            NLNT3=NLN*3
            IPL(NLNT3-2)=IP1
            IPL(NLNT3-1)=IPL(1)
            IPL(NLNT3)  =NTO
C - - DETERMINES THE VERTEX THAT DOES NOT LIE ON THE BORDER
C - - LINE SEGMENTS.
   62       ITT3=IT*3
            IPTI=IPT(ITT3-2)
            IF(IPTI.NE.IPL1.AND.IPTI.NE.IPL2)   GO TO 63
            IPTI=IPT(ITT3-1)
            IF(IPTI.NE.IPL1.AND.IPTI.NE.IPL2)   GO TO 63
            IPTI=IPT(ITT3)
C - - CHECKS IF THE EXCHANGE IS NECESSARY.
   63       IF(ICXCHG(XD,YD,IP1,IPTI,IPL1,IPL2).EQ.0)     GO TO 64
C - - MODIFIES THE IPT ARRAY WHEN NECESSARY.
            IPT(ITT3-2)=IPTI
            IPT(ITT3-1)=IPL1
            IPT(ITT3)  =IP1
            IPT(NTT3-1)=IPTI
            IF(JP2.EQ.JPMX)       IPL(JP2T3)=IT
            IF(JP2.EQ.NLO.AND.IPL(3).EQ.IT)     IPL(3)=NTO
C - - SETS FLAGS IN THE IWL ARRAY.
            JWL=JWL+4
            IWL(JWL-3)=IPL1
            IWL(JWL-2)=IPTI
            IWL(JWL-1)=IPTI
            IWL(JWL)  =IPL2
   64     CONTINUE
          NLO=NLN
          NLT3=NLNT3
          NLF=JWL/2
```

```
      IF(NLF.EQ.0)       GO TO 79
C - IMPROVES TRIANGULATION.
   70 NTT3P3=NTT3+3
      DO 78  IREP=1,NREP
        DO 76  ILF=1,NLF
          ILFT2=ILF*2
          IPL1=IWL(ILFT2-1)
          IPL2=IWL(ILFT2)
C - - LOCATES IN THE IPT ARRAY TWO TRIANGLES ON BOTH SIDES OF
C - - THE FLAGGED LINE SEGMENT.
          NTF=0
          DO 71  ITT3R=3,NTT3,3
            ITT3=NTT3P3-ITT3R
            IPT1=IPT(ITT3-2)
            IPT2=IPT(ITT3-1)
            IPT3=IPT(ITT3)
            IF(IPL1.NE.IPT1.AND.IPL1.NE.IPT2.AND.
     1         IPL1.NE.IPT3)       GO TO 71
            IF(IPL2.NE.IPT1.AND.IPL2.NE.IPT2.AND.
     1         IPL2.NE.IPT3)       GO TO 71
            NTF=NTF+1
            ITF(NTF)=ITT3/3
            IF(NTF.EQ.2)      GO TO 72
   71     CONTINUE
          IF(NTF.LT.2)        GO TO 76
C - - DETERMINES THE VERTEXES OF THE TRIANGLES THAT DO NOT LIE
C - - ON THE LINE SEGMENT.
   72     IT1T3=ITF(1)*3
          IPTI1=IPT(IT1T3-2)
          IF(IPTI1.NE.IPL1.AND.IPTI1.NE.IPL2)    GO TO 73
          IPTI1=IPT(IT1T3-1)
          IF(IPTI1.NE.IPL1.AND.IPTI1.NE.IPL2)    GO TO 73
          IPTI1=IPT(IT1T3)
   73     IT2T3=ITF(2)*3
          IPTI2=IPT(IT2T3-2)
          IF(IPTI2.NE.IPL1.AND.IPTI2.NE.IPL2)    GO TO 74
          IPTI2=IPT(IT2T3-1)
          IF(IPTI2.NE.IPL1.AND.IPTI2.NE.IPL2)    GO TO 74
          IPTI2=IPT(IT2T3)
C - - CHECKS IF THE EXCHANGE IS NECESSARY.
   74     IF(IDXCHG(XD,YD,IPTI1,IPTI2,IPL1,IPL2).EQ.0)
     1       GO TO 76
C - - MODIFIES THE IPT ARRAY WHEN NECESSARY.
          IPT(IT1T3-2)=IPTI1
          IPT(IT1T3-1)=IPTI2
          IPT(IT1T3)  =IPL1
          IPT(IT2T3-2)=IPTI2
          IPT(IT2T3-1)=IPTI1
```

```
                IPT(IT2T3)  =IPL2
C - - SETS NEW FLAGS.
                JWL=JWL+8
                IWL(JWL-7)=IPL1
                IWL(JWL-6)=IPTI1
                IWL(JWL-5)=IPTI1
                IWL(JWL-4)=IPL2
                IWL(JWL-3)=IPL2
                IWL(JWL-2)=IPTI2
                IWL(JWL-1)=IPTI2
                IWL(JWL)  =IPL1
                DO 75  JLT3=3,NLT3,3
                  IPLJ1=IPL(JLT3-2)
                  IPLJ2=IPL(JLT3-1)
                  IF((IPLJ1.EQ.IPL1.AND.IPLJ2.EQ.IPTI2).OR.
     1               (IPLJ2.EQ.IPL1.AND.IPLJ1.EQ.IPTI2))
     2                              IPL(JLT3)=ITF(1)
                  IF((IPLJ1.EQ.IPL2.AND.IPLJ2.EQ.IPTI1).OR.
     1               (IPLJ2.EQ.IPL2.AND.IPLJ1.EQ.IPTI1))
     2                              IPL(JLT3)=ITF(2)
   75           CONTINUE
   76        CONTINUE
             NLFC=NLF
             NLF=JWL/2
             IF(NLF.EQ.NLFC)       GO TO 79
C - - RESETS THE IWL ARRAY FOR THE NEXT ROUND.
             JWL=0
             JWL1MN=(NLFC+1)*2
             NLFT2=NLF*2
             DO 77  JWL1=JWL1MN,NLFT2,2
               JWL=JWL+2
               IWL(JWL-1)=IWL(JWL1-1)
               IWL(JWL)  =IWL(JWL1)
   77        CONTINUE
             NLF=JWL/2
   78     CONTINUE
   79 CONTINUE
C REARRANGES THE IPT ARRAY SO THAT THE VERTEXES OF EACH TRIANGLE
C ARE LISTED COUNTER-CLOCKWISE.
   80 DO 81   ITT3=3,NTT3,3
        IP1=IPT(ITT3-2)
        IP2=IPT(ITT3-1)
        IP3=IPT(ITT3)
        IF(SIDE(XD(IP1),YD(IP1),XD(IP2),YD(IP2),XD(IP3),YD(IP3))
     1       .GE.0.0)      GO TO 81
        IPT(ITT3-2)=IP2
        IPT(ITT3-1)=IP1
   81 CONTINUE
```

```
      NT=NTO
      NL=NLO
      RETURN
C ERROR EXIT
   90 WRITE (LUN,2090)  NDPO
      GO TO 93
   91 WRITE (LUN,2091)  NDPO,IP1,IP2,X1,Y1
      GO TO 93
   92 WRITE (LUN,2092)  NDPO
   93 WRITE (LUN,2093)
      NT=0
      RETURN
C FORMAT STATEMENTS
 2090 FORMAT(1X/23H ***   NDP LESS THAN 4./8H   NDP =,I5)
 2091 FORMAT(1X/29H ***   IDENTICAL DATA POINTS./
     1    8H   NDP =,I5,5X,5HIP1 =,I5,5X,5HIP2 =,I5,
     2    5X,4HXD =,E12.4,5X,4HYD =,E12.4)
 2092 FORMAT(1X/33H ***   ALL COLLINEAR DATA POINTS./
     1    8H   NDP =,I5)
 2093 FORMAT(35H ERROR DETECTED IN ROUTINE   IDTANG/)
      END
      FUNCTION  IDXCHG(X,Y,I1,I2,I3,I4)
C THIS FUNCTION DETERMINES WHETHER OR NOT THE EXCHANGE OF TWO
C TRIANGLES IS NECESSARY ON THE BASIS OF MAX-MIN-ANGLE CRITERION
C BY C. L. LAWSON.
C THE INPUT PARAMETERS ARE
C      X,Y =  ARRAYS CONTAINING THE COORDINATES OF THE DATA
C             POINTS,
C      I1,I2,I3,I4 = POINT NUMBERS OF FOUR POINTS P1, P2,
C             P3, AND P4 THAT FORM A QUADRILATERAL WITH P3
C             AND P4 CONNECTED DIAGONALLY.
C THIS FUNCTION RETURNS AN INTEGER VALUE 1 (ONE) WHEN AN EX-
C CHANGE IS NECESSARY, AND 0 (ZERO) OTHERWISE.
C DECLARATION STATEMENTS
      DIMENSION   X(15000),Y(15000)
      EQUIVALENCE (C2SQ,C1SQ),(A3SQ,B2SQ),(B3SQ,A1SQ),
     1            (A4SQ,B1SQ),(B4SQ,A2SQ),(C4SQ,C3SQ)
C PRELIMINARY PROCESSING
   10 X1=X(I1)
      Y1=Y(I1)
      X2=X(I2)
      Y2=Y(I2)
      X3=X(I3)
      Y3=Y(I3)
      X4=X(I4)
      Y4=Y(I4)
C CALCULATION
   20 IDX=0
```

```
      U3=(Y2-Y3)*(X1-X3)-(X2-X3)*(Y1-Y3)
      U4=(Y1-Y4)*(X2-X4)-(X1-X4)*(Y2-Y4)
      IF(U3*U4.LE.0.0)     GO TO 30
      U1=(Y3-Y1)*(X4-X1)-(X3-X1)*(Y4-Y1)
      U2=(Y4-Y2)*(X3-X2)-(X4-X2)*(Y3-Y2)
      A1SQ=(X1-X3)**2+(Y1-Y3)**2
      B1SQ=(X4-X1)**2+(Y4-Y1)**2
      C1SQ=(X3-X4)**2+(Y3-Y4)**2
      A2SQ=(X2-X4)**2+(Y2-Y4)**2
      B2SQ=(X3-X2)**2+(Y3-Y2)**2
      C3SQ=(X2-X1)**2+(Y2-Y1)**2
      S1SQ=U1*U1/(C1SQ*AMAX1(A1SQ,B1SQ))
      S2SQ=U2*U2/(C2SQ*AMAX1(A2SQ,B2SQ))
      S3SQ=U3*U3/(C3SQ*AMAX1(A3SQ,B3SQ))
      S4SQ=U4*U4/(C4SQ*AMAX1(A4SQ,B4SQ))
      IF(AMIN1(S1SQ,S2SQ).LT.AMIN1(S3SQ,S4SQ))      IDX=1
   30 IDXCHG=IDX
      RETURN
      END
```

```
C PROGRAM FRA4M.FOR
C
C OWNER CESAR TADEU DE MIRANDA
C DATE 04/OCT/83
C
C THIS IS A FORTRAN PROGRAM THAT IMPLEMENTS THE FRANKE'S THIN
C PLATE SPLINE ALGORITHM FOR INTERPOLATION ON A SCATTERED SET
C OF DATA.
C
C INPUT PARAMETERS
C     FILE FOR012.DAT: FILE CONTAINING THE PARAMETER "NPPR" THAT IS
C     AN ESTIMATE OF THE AVERAGE NUMBER OF POINT PER RECTANGLE
C     WHERE THE THIN PLATES WILL BE EVALUEATED.
C     FILE FOR019.DAT: FILE CONTAINING THE NUMBER OF CONTOURS PER
C     EACH LEVEL AND THE GRAY LEVEL VALUE OF EACH LEVEL.
C     FILE FOR023.DAT: FILE CONTAINING THE SELECTED KNOTS REPRESENTING
C     THE IMAGE AS A SCATTERED SET OF DATA.
C
C OUTPUT PARAMETERS
C     FILE FOR031.DAT: UNIFORM GRID (256,256) REPRESENTING THE RECONS-
C     TRUCTED IMAGE.
C
C VARIABLE DECLARATION
      REAL X(15000),Y(15000),F(15000),XO(256),YO(256),FO(256,256)
      REAL WK(120000),CLEV
      INTEGER L,K,NCONT,I,N,IX,IY,ITYPE,IER,TCONT,IWK(110000)
      INTEGER MODE,NPWR,NNF,NXO,NYO,J,IMIN,IMAX,ITEMP
      INTEGER MX(15000),MY(15000),KT,FLAG,II,IPROV(256),IST,IEND
      REAL RNW,RNF
      REAL GI,GA
C
C ERROR CODE
C     ERROR: 0 NORMAL TERMINATION
C           -1 DIMENSION IS NOT ENOUGH
C            POSITIVE VALUES OF THE ERROR ARE GENERATED BY THE
C            SUBROUTINE QSHEP, SEE DESCRIPTION OF THESE ERRORS
C            IMBED IN THAT BLOCK.
C ENTER THE DATA
      L=0
      K=1
      IER=0
      TCONT=0
      TYPE *,'ENTER UNIT NUMBER OF CONTOURS IR'
C      ACCEPT *,IR(TO RUN ON BATCH)
      IR=23
      READ(19,100) NCONT,CLEV
      GI=CLEV
      DO WHILE ((NCONT.NE.-3).AND.(IER.EQ.0))
```

```
            TYPE *,'LEVEL # ',K
            TYPE *,' '
            ICONT=ICONT+NCONT
            IF(NCONT.NE.0) THEN
              TYPE *,'# OF CONTOURS IN THIS LEVEL:',NCONT,'   VALUE',CLEV
              DO I=1,NCONT
                READ(IR,124) IX,IY
C               TYPE *,' IX',IX,'   IY',IY
                N=IX
                ITYPE=IY
                DO J=1,N
                  READ(IR,124)IX,IY
                  L=L+1
                  IF(L.GT.15000) THEN
                    IER=-1
                    TYPE *,'ERROR   ',IER
                    TYPE *,'# OF CONTOURS OF INPUT EXCEEDS THE DIMENSION
     1              OF ARRAYS MX, MY,F'
                    STOP
                  END IF
                  MX(L)=IX
                  MY(L)=IY
                  F(L)=CLEV
                END DO
              END DO
            ELSE
              TYPE *,'THE LEVEL',K,' HAS NO CONTOURS'
            END IF
            K=K+1
            GA=CLEV
            READ(IS,100)NCONT,CLEV
          END DO
C
C ELIMINATE THE REPEATED ORDINATE PAIRS (MX,MY)
          X(1)=FLOAT(MX(1))
          Y(1)=FLOAT(MY(1))
          KT=1
          DO I=2,L
            FLAG=0
            I1=I-1
            DO J=I1,1,-1
              IF((MX(I).EQ.MX(J)).AND.(MY(I).EQ.MY(J))) THEN
                FLAG=1
C     TYPE *,'TRAP A ALLIEN'
                GO TO 20
              END IF
            END DO
 20         CONTINUE
```

```
            IF(FLAG.EQ.0) THEN
              KT=KT+1
              X(KT)=FLOAT(MX(I))
              Y(KT)=FLOAT(MY(I))
              F(KT)=F(I)
            END IF
          END DO
          TYPE *,'TOTAL NUMBER OF LEVELS',K-1
          TYPE *,'TOTAL NUMBER OF CONTOURS',TCONT
          TYPE *,'TOTAL NUMBER OF POINTS',L
          TYPE *,'TOTAL NUMBER OF POINTS WITHOUT REPETITION',KT
          TYPE *,'LOWER BOUND:',GI,'   UPPER BOUND:',GA
C
C ADDING THE FRANKE PARAMETERS
          MODE=1
C INPUT NPPR FROM FILE FOR012.DAT
          READ(12,150)NPPR
          TYPE *,'NPPR',NPPR
          NPI=KT
C DEFINE SIZE OF THE OUTPUT GRID AND DIMENSION OF WORK ARRAYS TO
C BE PASSED TO THE SUBROUTINES.
          NXO=256
          NYO=256
          NIWK=110000
          NWK=120000
          DO I=1,256
            XO(I)=FLOAT(I)
            YO(I)=XO(I)
          END DO
          TYPE *,'INVOKE FRANKE ALGORITHNM'
          CALL LCTPS(MODE,NPPR,NPI,X,Y,F,NXO,XO,NYO,YO,IWK,NIWK,NIWKU,WK,
         1 NWK,NWKU,FO,IER)
          TYPE *,'FINISH FRANKE ALGORITHM'
          TYPE *,'ERROR',IER,'  NIWKU',NIWKU,'  NWKU',NWKU
C
C FIND MAX AND MIN OF THE RECONSTRUCTED UNIFORM GRID "FO"
          IMIN=NINT(FO(1,1))
          IMAX=NINT(FO(1,1))
          DO I=1,256
            DO J=1,256
              ITEMP=NINT(FO(I,J))
              IF(ITEMP.GT.IMAX) THEN
                IMAX=ITEMP
              ELSE IF (ITEMP.LT.IMIN) THEN
                IMIN=ITEMP
              END IF
            END DO
          END DO
```

```
      TYPE *,'IMAX',IMAX
      TYPE *,'IMIN',IMIN
      IRANGE=IMAX-IMIN
      TYPE *,'RANGE OF GRAY LEVEL',IRANGE
C
C
C NORMALIZATION
      TYPE *,'START THE NORMALIZATION'
      DO I=1,256
        DO J=1,256
          IF(FO(I,J).LT.GI) THEN
            FC(I,J)=GI
          ELSE IF(FO(I,J).GT.GA) THEN
            FC(I,J)=GA
          END IF
        END DO
      END DO
      TYPE *,'END NORMALIZATION'
C
C STORE THE MATRIX IN A FILE 31
       DO I=1,256
          WRITE(31,130) I
          DO J=1,256
            IPROV(J)=NINT(FO(I,J))
          END DO
          DO K=0,7
            IST=32*K +1
            IEND=IST+31
            WRITE(31,128)(IPROV(J),J=IST,IEND)
          END DO
       END DO
      TYPE *,'END OF PROGRAM'
      STOP
100   FORMAT(I5,F10.5)
124   FORMAT(2I5)
128   FORMAT(32I4)
130   FORMAT(I4)
150   FORMAT(I2)
      END
```

```
      SUBROUTINE LOTPS (MODE,NPPR,NPI,XI,YI,FI,NXO,XO,NYO,YO,IWK,NIWK,
     1 NIWKU,WK,NWK,NWKU,FO,KER)
C
C     DATE OF LAST CHANGE/CORRECTION   7 DECMEMBER 1982
C
C     THIS SUBROUTINE SERVES AS A USER INTERFACE TO THE SET OF
C     SUBROUTINES THAT IMPLEMENT FRANKE'S METHOD OF SURFACE INTERPO-
C     LATION.  RECTANGULAR REGIONS ARE USED WITH PRODUCT CUBIC
C     HERMITE WEIGHT FUNCTIONS.  THE RECTANGLES ARE CHOSEN IN AN ATTEMPT
C     TO OBTAIN ABOUT NPPR POINTS IN EACH REGION.  THE SAME NUMBER OF
C     GRID LINES IS USED IN EACH DIRECTION.  LOCAL INTERPOLATION FUNC-
C     TIONS ARE THE THIN PLATE SPLINES DESCRIBED BY DUCHON AND OTHERS.
C     A DESCRIPTION OF THE METHOD AND REFERENCES APPEAR IN
C
C     SMOOTH INTERPOLATION OF SCATTERED DATA BY LOCAL THIN PLATE SPLINES
C           NAVAL POSTGRADUATE SCHOOL REPORT  NPS-53-81-002
C              RICHARD FRANKE
C              DEPARTMENT OF MATHEMATICS
C              NAVAL POSTGRADUATE SCHOOL
C              MONTEREY, CALIFORNIA  93940
C                   (408)646-2758 / 2206
C
C     THIS HAS ALSO APPEARED IN
C           COMPUTERS AND MATHEMATICS WITH APPLICATIONS 8(1982)273-281
C
C        DIFFICULTIES AND SUCCESSES WITH THIS PROGRAM SHOULD BE
C        COMMUNICATED TO THE AUTHOR.
C
C     THE ARGUMENTS ARE AS FOLLOWS.
C
C        MODE - INPUT.  INDICATES THE STATUS OF THE CALCULATION.
C                     = 1,  SET UP THE PROBLEM.  COMPUTE THE COEFFICIENTS
C                           FOR THE LOCAL APPROXIMATIONS BY THIN PLATE
C                           SPLINES, AND RETURN THE GRID OF INTERPOLATED
C                           FUNCTION VALUES INDICATED BY NXO, XO, NYO, YO
C                           IN FO.
C                     = 2,  THE PROBLEM HAS BEEN SET UP PREVIOUSLY.  CALCU-
C                           LATE THE GRID OF INTERPOLATED POINTS INDICATED
C                           BY NXO, XO, NYO, YO IN FO.  THE PROGRAM ASSUMES
C                           THAT THE ARRAYS XI, YI, IWK, AND WK ARE
C                           UNCHANGED FROM THE PREVIOUS CALL.
C        NPPR - INPUT.  DESIRED AVERAGE NUMBER OF POINTS PER REGION.
C                       THE SUGGESTED VALUE IS TEN.  SHOULD BE AT LEAST
C                       FOUR.  VALUES LARGER THAN FIFTEEN COULD REQUIRE
C                       MINOR PROGRAM MODIFICATIONS TO ALLOW MORE
C                       STORAGE FOR THE EQUATION SOLVER.  THIS DEPENDS
C                       ON THE DISPOSITION OF THE POINTS.
C                       IF THE USER WISHES TO SPECIFY HIS OWN GRID LINES
```

```
C                 X TILDA AND Y TILDA, HE MAY DO SO BY SETTING
C                 NPPR = 0 AND SETTING NECESSARY VALUES IN THE
C                 ARRAYS IWK AND WK, AS NOTED BELOW.  DATA WHICH
C                 HAS A POOR DISTRIBUTION OVER THE REGION OF
C                 INTEREST SHOULD PROBABLY HAVE THE GRID SPECIFIED
C                 THIS IS ALSO ADVISABLE IF THE X-Y POINTS OCCUR
C                 ON LINES.
C     NPI   - INPUT.  NUMBER OF INPUT DATA POINTS.
C     XI    - -
C     YI    - INPUT.  THE DATA POINTS (XI,YI,FI), I=1,...,NPI.
C     FI    - /
C     NXO   - INPUT.  THE NUMBER OF XO VALUES AT WHICH THE INTERP-
C                     OLATION FUNCTION IS TO BE CALCULATED.
C     XO    - INPUT.  THE VALUES OF X AT WHICH THE INTERPOLATION
C                     FUNCTION IS TO BE CALCULATED.  THESE SHOULD
C                     BE IN INCREASING ORDER FOR MOST EFFICIENT
C                     EVALUATION.
C     NYO   - INPUT.  THE NUMBER OF YO VALUES AT WHICH THE INTERP-
C                     OLATION FUNCTION IS TO BE CALCULATED.
C     YO    - INPUT.  THE VALUES OF Y AT WHICH THE INTERPOLATION
C                     FUNCTION IS TO BE CALCULATED.  THESE SHOULD
C                     BE IN INCREASING ORDER FOR MOST EFFICIENT
C                     EVALUATION.
C     IWK   - INPUT AND OUTPUT.  THIS ARRAY IS OUTPUT WHEN MODE = 1
C                     AND IS INPUT WHEN MODE = 2.  THIS MUST BE
C                     AN ARRAY DIMENSIONED APPROXIMATELY 6*NPI.
C                     WHEN NPPR IS INPUT AS ZERO THE USER MUST
C                     SPECIFY THE NUMBER OF VERTICAL GRID LINES (THE
C                     NUMBER OF X TILDA VALUES) IN IWK(1) AND THE
C                     NUMBER OF HORIZONTAL GRID LINES (THE NUMBER OF
C                     Y TILDA VALUES) IN IWK(2).
C     NIWK  - INPUT.  ON ENTRY WITH MODE = 1 THIS MUST BE SET TO THE
C                     DIMENSION OF THE ARRAY IWK.
C     NIWKL - OUTPUT.  THE ACTUAL NUMBER OF LOCATIONS USED IN THE
C                     ARRAY IWK.
C     WK    - INPUT AND OUTPUT.  THIS ARRAY IS OUTPUT WHEN MODE = 1
C                     AND IS INPUT WHEN MODE = 2.  THIS MUST BE AN
C                     ARRAY DIMENSIONED APPROXIMATELY 7*NPI LOCATIONS.
C                     LOCATIONS.
C                     WHEN NPPR IS INPUT AS ZERO THE USER MUST SPECIFY
C                     THE VALUES OF X TILDA AND Y TILDA AS FOLLOWS.
C                     WK(2), ... , WK(NXG+1) ARE THE NXG (= IWK(1))
C                     X GRID VALUES, X(I) TILDA, IN INCREASING ORDER.
C                     TYPICALLY WK(1) = MIN X(I), ALTHOUGH IT NEED
C                     NOT BE.  WK(1) MUST BE LESS THAN OR EQUAL TO
C                     WK(2), AND SHOULD BE LESS THAN OR EQUAL TO
C                     MIN X(I).  WK(NXG+2) IS USUALLY MAX X(I), AL-
C                     THOUGH IT NEED NOT BE.  WK(NXG+2) MUST BE
```

```
C                       GREATER THAN WK(NXG+1), AND SHOULD BE GREATER
C                       THAN OR EQUAL TO MAX X(I).
C                       THE VALUES OF WK(NXG+3), ... , WK(NXG+NYG+4)
C                       ARE THE Y GRID VALUES, Y(I) TILDA, AND MUST
C                       SATISFY DUAL CONDITIONS.
C     NWK   - INPUT.    ON ENTRY WITH MODE = 1 THIS MUST BE SET TO THE
C                       DIMENSION OF THE ARRAY WK.
C     NWKU  - OUTPUT.   THE ACTUAL NUMBER OF LOCATIONS USED IN THE
C                       ARRAY WK.
C     FO    - OUTPUT.   VALUES OF THE INTERPOLATION FUNCTION AT THE
C                       GRID OF POINTS INDICATED BY NXO, XO, NYO, YO.
C                       FO IS ASSUMED TO BE DIMENSIONED (NXO,NYO) IN THE
C                       CALLING PROGRAM.
C     KER   - OUTPUT.   RETURN INDICATOR.
C                 = 0,  NORMAL RETURN.
C                 =-1,  PROBLEM HAS NOT BEEN PREVIOUSLY SET UP (CLOTPS
C                       CALLED WITH MODE = 1)
C                 = 1,  ERROR RETURN FROM CLOTPS, SINGULAR MATRIX IN THE
C                       CALCULATION OF THE THIN PLATE SPLINES.
C                 = 2,  ERROR RETURN FROM CLOTPS.  SOME RECTANGLE (I,J)
C                       HAS MORE THAN THE ALLOWED NUMBER OF POINTS
C                       ASSOCIATED WITH IT.  SEE CLOTPS FOR THE FIX.
C                 = 3,  PREVIOUS ERROR RETURN FROM CLOTPS HAS NOT BEEN
C                       CORRECTED.
C                 = 4,  IWK AND/OR WK ARRAYS HAVE NOT BEEN DIMENSIONED
C                       LARGE ENOUGH IN THE CALLING PROGRAM.  REDIMEN-
C                       SION IWK AND WK TO AT LEAST THE SIZE INDICATED
C                       BY NIWKU AND NWKU, RESPECTIVELY.
C                 = 5,  MODE IS OUT OF RANGE.
C
C     SUBPROGRAMS CALLED BY THIS ROUTINE ARE    GRID, LOCLIP, CLOTPS,
C     AND EVLTPS.  ALSO REQUIRED ARE  VSRTA  FROM THE IMSL LIBRARY AND
C     THE ROUTINES  DECOMP AND SOLVE  FROM FORSYTHE, MALCOLM, AND MOLER.
C
      DIMENSION XI(1), YI(1), FI(1), IWK(1), WK(1), XO(1), YO(1),
     1 FO(NXO,1)
      DATA KER/-1/
      IF (MODE.LT.1.OR.MODE.GT.2) GO TO 220
      KER = 0
C
C     ON INITIAL ENTRY MODE = 1, THE GRID LINES ARE SET UP,
C     LOCAL INTERPOLATION POINTS ARE DETERMINED AND LOCAL APPROXIMATIONS
C     ARE COMPUTED.
C
      IF (MODE.EQ.2) GO TO 140
      NXGWK = 1
      NPWK = 3
      IF (NPPR.LE.0) GO TO 100
```

```
      NXG = SQRT(4.*FLOAT(NPI)/FLOAT(NPPR))-.5
      NXG = MAX0(NXG,1)
      NYG = NXG
      IWK(1) = NXG
      IWK(2) = NYG
      GO TO 120
  100 NXG = IWK(1)
      NYG = IWK(2)
  120 IALWK = NXG*NYG+5
      IABWK = IALWK + 3*NXG*NYG
      NYGWK = NXG+3
      MPWK = NXG*NYG+4
C
      IF(NPPR.LE.0)GO TO 130
      CALL GRID(XI,NPI,NXG,WK(NXGWK),WK(IALWK))
      CALL GRID(YI,NPI,NYG,WK(NYGWK),WK(IALWK))
  130 CONTINUE
C
C     DETERMINE THE LOCAL INTERPOLATION POINTS FOR THE REGIONS.
      MWK = NWK - MPWK + 1
      CALL LCCLIP (NXG,WK(NXGWK),NYG,WK(NYGWK),NPI,XI,YI,IWK(NPWK),
     1IWK(MPWK),MWK,WK(IALWK))
      NWKU = IABWK +IWK(MPWK - 1)-2
      NIWKU = NXG*NYG+2+IWK(MPWK-1)
      IF (NIWKU.GT.NIWK) GO TO 200
      IF (NWKU.GT.NWK) GO TO 200
C
C     COMPUTE THE LOCAL APPROXIMATIONS.
      CALL CLCTPS (XI,YI,FI,NXG,WK(NXGWK),NYG,WK(NYGWK),IWK(NPWK),IWK
     1(MPWK),WK(IALWK),WK(IABWK),IER)
      KERU = IER
      IF (IER.NE.0) GO TO 160
  140 IF (KERU.NE.0) GO TO 180
C
C     COMPUTE THE FUNCTION VALUES ON THE DESIRED GRID OF POINTS.
C
      CALL EVLTPS (XI,YI,IWK(1),WK(NXGWK),IWK(2),WK(NYGWK),IWK(NPWK),
     1 IWK(MPWK),WK(IALWK),WK(IABWK),NXU,XU,NYU,YU,FU)
      RETURN
C
C     ERROR RETURNS
C
  160 KER = IER
      RETURN
  180 KER = 3
      IF (KERU.LT.0) KER = -1
      RETURN
  200 KER = 4
```

```
      RETURN
  220 KER = 5
      RETURN
      END
      SUBROUTINE CLOTPS (XI,YI,FI,NXG,XG,NYG,YG,NP,MP,AL,AB,IER)
C
C     THIS SUBROUTINE CONSTRUCTS THE LOCAL APPROXIMANTS FOR THE GRID
C     VERSION OF FRANKE'S METHOD.  THE LOCAL APPROXIMATIONS ARE TAKEN
C     TO BE THE THIN PLATE SPLINES DESCRIBED BY DUCHON AND OTHERS.
C
C     THE ARGUMENTS ARE AS FOLLOWS.
C
C        XI    - -
C        YI    - INPUT.  THE DATA POINTS (XI,YI,FI),I=1,NPI.
C        FI    - /
C        NXG   - INPUT.  THE NUMBER OF VERTICAL GRID LINES.
C        XG    - INPUT.  THE COORDINATES OF THE VERTICAL GRID LINES, IN
C                        INCREASING ORDER.
C        NYG   - INPUT.  THE NUMBER OF HORIZONTAL GRID LINES.
C        YG    - INPUT.  THE COORDINATES OF THE HORIZONTAL GRID LINES, IN
C                        INCREASING ORDER.
C        NP    - INPUT.  AN ARRAY WHICH GIVES THE INITIAL SUBSCRIPT IN
C                        THE ARRAY MP AT WHICH THE SUBSCRIPTS FOR THE
C                        LOCAL INTERPOLATION POINTS ARE STORED.
C        MP    - INPUT.  AN ARRAY WHICH GIVES THE SUBSCRIPTS FOR THE
C                        LOCAL INTERPOLATION POINTS.
C        AL    - OUTPUT.  THE COEFFICIENTS FOR THE LINEAR PART OF THE
C                        LOCAL THIN PLATE SPLINE FIT.
C        AB    - OUTPUT.  THE COEFFICIENTS FOR THE THIN PLATE SPLINES
C        IER   - OUTPUT.  RETURN INDICATOR.
C                  = 0,  NORMAL RETURN.
C                  = 1,  SINGULAR MATRIX HAS BEEN DETECTED IN THE
C                        THIN PLATE SPLINE FIT.
C                             IN CASE OF A SINGULAR MATRIX, THE GRID
C                             VALUE (I,J) AND THE DATA POINTS ASSOCIATED
C                             WITH THAT RECTANGLE ARE PRINTED.
C                  = 2,  THE NUMBER OF POINTS ASSOCIATED WITH SOME
C                        RECTANGLE  I,J  IS BIGGER THAN PRESENTLY PERMIT-
C                        TED.  THE ARRAY C MUST BE DIMENSIONED (NC,NC+3)
C                        LOCATIONS.  SEE BELOW FOR DETAILS.
C                        THE ROUTINES FOR SOLUTION OF LINEAR SYSTEMS,
C                        DECOMP AND SOLVE, FROM FORSYTHE, MALCOLM AND
C                        MOLER IS USED.
C
      DIMENSION XI(1), YI(1), FI(1), NP(1), MP(1), AL(1), AB(1), XG(1),
     1 YG(1)
C
C     THE ARRAY C MUST BE DIMENSIONED (NC,NC+3) LOCATIONS, AND
```

```
C       MUST AGREE WITH NC AS DECLARED IN THE DATA STATEMENT.  THIS
C       ARRAY PROVIDES STORAGE FOR THE LINEAR SYSTEM OF EQUATIONS
C       FOR THE COEFFICIENTS IN THE LOCAL APPROXIMANTS, AS WELL
C       AS SCRATCH STORAGE REQUIRED BY DECOMP AND SOLVE.  THE NUMBER
C       OF POINTS ASSOCIATED WITH ANY RECTANGLE MUST NOT EXCEED NC-3.
C
        DIMENSION C(133,136)
        DATA NCOT,NC/6,133/
C
C       ARITHMETIC STATEMENT FUNCTION FOR THE THIN PLATE SPLINE BASIS
C       FUNCTIONS.
C
        PHI(X,Y,XI,YI) = ((X-XI)**2+(Y-YI)**2)*ALOG(((X-XI)**2+(Y-YI)**2)
       1 + 1.E-20)
        IER = 0
        IJ = 0
        NCM3 = NC - 3
        DO 160 J=1,NYG
        DO 140 I=1,NXG
        IJ = IJ + 1
        LEND = NP(IJ+1) - NP(IJ)
        IF(LEND.GT.NCM3)GO TO 360
  140 CONTINUE
  160 CONTINUE
        IJ = 0
        NC1 = NC + 1
        NC2 = NC1 + 1
        NC3 = NC2 + 1
C
        DO 260 J=1,NYG
        DY = YG(J+2)-YG(J)
C
        DO 240 I=1,NXG
        DX = XG(I+2)-XG(I)
        IJ = IJ+1
        LEND = NP(IJ+1)-NP(IJ)
        LEND3 = LEND + 3
        IALS = (IJ-1)*3
C
        DO 200 LI=1,LEND
        MPI = NP(IJ)+LI-1
        KI = MP(MPI)
        XKI = (XI(KI)-XG(I))/DX
        YKI = (YI(KI)-YG(J))/DY
        C(LEND+1,LI) = 1.
        C(LEND+2,LI) = XKI
        C(LEND+3,LI) = YKI
        C(LI,LEND+1) = 1.
```

```
      C(LI,LEND+2) = XKI
      C(LI,LEND+3) = YKI
C
      DO 180 LJ=1,LI
      MPJ = NP(IJ)+LJ-1
      KJ = MP(MPJ)
      XKJ = (XI(KJ)-XG(I))/DX
      YKJ = (YI(KJ)-YG(J))/DY
      C(LI,LJ) = PHI(XKI,YKI,XKJ,YKJ)
  180 C(LJ,LI) = C(LI,LJ)
C
      C(LI,NC1) = FI(KI)
  200 CONTINUE
      DO 215 LLI=1,3
      LI = LEND + LLI
      DO 210 LLJ = 1,3
      LJ = LEND + LLJ
  210 C(LI,LJ) = 0.
      C(LI,NC1) = 0.
  215 CONTINUE
C
      CALL DECOMP(NC,LEND3,C,COND,C(1,NC2),C(1,NC3))
      IF((COND+1.).EQ.COND)GO TO 300
      CALL SOLVE(NC,LEND3,C,C(1,NC1),C(1,NC2))
C
      DO 220 LI=1,LEND
      IAB = NP(IJ)+LI-1
  220 AB(IAB) = C(LI,NC1)
C
      AL(IALS+1) = C(LEND+1,NC1)
      AL(IALS+2) = C(LEND+2,NC1)
      AL(IALS+3) = C(LEND+3,NC1)
  240 CONTINUE
C
  260 CONTINUE
C
      RETURN
C
C     ERROR RETURNS
C
  300 IER = 1
  320 WRITE (NOUT,1) I,J,IER
      MPST = NP(IJ)
      MPSE = MPST+LEND-1
C
      DO 340 MPS=MPST,MPSE
      I = MP(MPS)
  340 WRITE (NOUT,2) I,XI(I),YI(I),FI(I)
```

```
C
      RETURN
  360 IER = 2
      WRITE (NOUT,3) I,J,LEND,NCM3
      RETURN
C
    1 FORMAT (46HOSINGULAR MATRIX DETECTED FOR GRID POINT I,J =,2I5,12H
     1,    IER = ,I2//40H THE DATA POINTS INVOLVED ARE AS FOLLOWS/)
    2 FORMAT (1X,I5,3E15.6)
    3 FORMAT (15HOTHE RECTANGLE ,I2,1H,,I3,5H HAS ,I3,27H POINTS ASSOCIA
     1TED WITH IT /51H THE CURRENT LIMIT IMPOSED IN SUBROUTINE CLOTPS IS
     2 ,I3/84H THIS MAY BE MODIFIED BY REDEFINING NC AND CHANGING THE DI
     3MENSION OF C APPROPRIATELY)
      END
      SUBROUTINE LOCLIP (NXG,XG,NYG,YG,NPI,XI,YI,NP,MP,MPM,D)
C
CCC   THIS SUBROUTINE DETERMINES THE LOCAL INTERPOLATION POINTS FOR THE
CCC   GRID VERSION OF FRANKE'S METHOD OF SURFACE INTERPOLATION.
CCC   MINPTS POINTS ARE REQUIRED FOR EACH REGION.
CCC   IF FEWER THAN MINPTS POINTS ARE FOUND IN THE REGION, THE NEXT
CCC   CLOSEST POINTS (IN THE SUP NORM AFTER THE CURRENT RECTANGLE IS
CCC   TRANSFORMED ONTO (0,1)) ARE USED.  MINPTS IS SET TO 3, WHICH IS
CCC   THE RECOMMENDED VALUE, ALTHOUGH IT MAY BE ALTERED.
CCC
CCC   THE ARGUMENTS ARE AS FOLLOWS.
CCC
CCC      NXG   - INPUT.  NUMBER OF VERTICAL GRID LINES.
CCC      XG    - INPUT.  THE COORDINATES OF THE VERTICAL GRID LINES, IN
CCC                       INCREASING ORDER
CCC      NYG   - INPUT.  NUMBER OF HORIZONTAL GRID LINES.
CCC      YG    - INPUT.  THE COORDINATES OF THE HORIZONTAL GRID LINES,
CCC                       IN INCREASING ORDER.
CCC      NPI   - INPUT.  THE NUMBER OF DATA POINTS.
CCC      XI    - -
CCC      YI    - INPUT.  THE DATA POINTS (XI,YI), I=1,...,NPI.
CCC      FI    - /
CCC      NP    - OUTPUT.  AN ARRAY WHICH GIVES THE INITIAL SUBSCRIPT IN
CCC                       THE ARRAY MP AT WHICH THE SUBSCRIPTS FOR THE
CCC                       LOCAL INTERPOLATION POINTS ARE STORED.
CCC      MP    - OUTPUT.  AN ARRAY WHICH GIVES THE SUBSCRIPTS FOR THE
CCC                       LOCAL INTERPOLATION POINTS.
CCC      MPM   - INPUT.  DIMENSION OF THE ARRAY MP IN THE CALLING PROGRAM
CCC      D     - A WORK ARRAY OF DIMENSION AT LEAST NPI.
C
      DIMENSION XG(1), YG(1), XI(1), YI(1), NP(1), MP(1), D(1)
      DATA MINPTS/3/
      IJ = 1
      NP(1) = 1
```

```
      L = 0
C
      DO 200 J=1,NYG
      YGA = (YG(J+2)+YG(J))/2.
      DYG = YG(J+2)-YG(J)
C
      DO 180 I=1,NXG
      XGA = (XG(I+2)+XG(I))/2.
      DXG = XG(I+2)-XG(I)
      IJ = IJ+1
C
C     DETERMINE THE POINTS IN THE (I,J)TH RECTANGLE.
C
      DO 120 NK=1,NPI
      D(NK) = AMAX1(ABS(XI(NK) - XGA)/DXG,ABS(YI(NK) - YGA)/DYG)
      IF(D(NK).GT..6125)GO TO 120
      D(NK) = 1.E10
      L = L + 1
      LL = MINO(L,MPM)
      MP(LL) = NK
  120 CONTINUE
C
      NP(IJ) = L+1
      IF (NP(IJ)-NP(IJ-1).GE.MINPTS) GO TO 180
C
C     ADD THE CLOSEST POINTS IF THERE ARE LESS THAN MINPTS IN THE
C     RECTANGLE.
C
      LM = MINPTS-(NP(IJ)-NP(IJ-1))
C
      DO 160 II=1,LM
      L = L+1
      LL = MINO(L,MPM)
      MP(LL) = 1
      DM = D(1)
C
      DO 140 NK=2,NPI
      IF (D(NK).GE.DM) GO TO 140
      DM = D(NK)
      MP(LL) = NK
  140 CONTINUE
C
      NK = MP(LL)
  160 D(NK) = 1.E10
C
      NP(IJ) = L+1
  180 CONTINUE
C
```

```
  200 CONTINUE
C
      RETURN
      END
      SUBROUTINE EVLTPS (XI,YI,NXG,XG,NYG,YG,NP,MP,AL,AB,NXO,XO,NYO
     1,YO,FO)
C
C     THIS SUBROUTINE EVALUATES THE INTERPOLANT FOR THE GRID VERSION OF
C     FRANKE'S METHOD.  THE FUNCTION IS EVALUATED AT THE GRID OF POINTS
C     INDICATED BY NXO, XO, NYO, YO, AND THESE VALUES ARE RETURNED
C     IN THE ARRAY FO, WHICH IS ASSUMED TO BE DIMENSIONED (NXO,NYO).
C
C     THE ARGUMENTS ARE AS FOLLOWS.
C
C        XI    - -
C        YI    - INPUT.  THE DATA POINTS (XI,YI,FI),I=1,...,NPI.
C        FI    - /
C        NXG   - INPUT.  THE NUMBER OF VERTICAL GRID LINES.
C        XG    - INPUT.  THE COORDINATES OF THE VERTICAL GRID LINES, IN
C                        INCREASING ORDER.
C        NYG   - INPUT.  THE NUMBER OF HORIZONTAL GRID LINES.
C        YG    - INPUT.  THE COORDINATES OF THE HORIZONTAL GRID LINES,
C                        IN INCREASING ORDER.
C        NP    - INPUT.  AN ARRAY WHICH GIVES THE INITIAL SUBSCRIPT IN
C                        THE ARRAY MP AT WHICH THE SUBSCRIPTS FOR THE
C                        LOCAL INTERPOLATION POINTS ARE STORED.
C        MP    - INPUT.  AN ARRAY WHICH GIVES THE SUBSCRIPTS FOR THE
C                        LOCAL INTERPOLATION POINTS.
C        AL    - INPUT.  THE COEFFICIENTS FOR THE LINEAR PART OF THE
C                        THIN PLATE SPLINE APPROXIMATIONS.
C        AB    - INPUT.  THE COEFFICIENTS FOR THE LOCAL THIN PLATE
C                        SPLINE APPROXIMATIONS.
C        NXO   - INPUT.  THE NUMBER OF XO VALUES AT WHICH THE INTERPO-
C                        LATION FUNCTION IS TO BE CALCULATED.
C        XO    - INPUT.  THE VALUES OF X AT WHICH THE INTERPOLATION
C                        FUNCTION IS TO BE CALCULATED.
C        NYO   - INPUT.  THE NUMBER OF YO VALUES AT WHICH THE INTERPO-
C                        LATION FUNCTION IS TO BE CALCULATED.
C        YO    - INPUT.  THE VALUES OF Y AT WHICH THE INTERPOLATION
C                        FUNCTION IS TO BE CALCULATED.
C        FO    - OUTPUT.  VALUES OF THE INTERPOLATION FUNCTION AT THE
C                        GRID POINTS INDICATED BY NXO, XO, NYO, YO.
C                        FO IS ASSUMED TO BE DIMENSIONED (NXO,NYO) IN THE
C                        CALLING PROGRAM.
C
      DIMENSION XG(1), YG(1), XI(1), YI(1), NP(1), MP(1), FO(4), AL(1),
     1AB(1), XO(1), YO(1), FO(NXO,1)
C
```

```
C       ARITHMETIC STATEMENT FUNCTION FOR THE HERMITE CUBIC.
C
        H3(S) = 1. - S**2*(3. - 2.*S)
C
C       ARITHMETIC STATEMENT FUNCTION FOR THE THIN PLATE SPLINE BASIS
C       FUNCTIONS.
C
        PHI(X,Y,XI,YI) = ((X-XI)**2+(Y-YI)**2)*ALOG(((X-XI)**2+(Y-YI)**2)
       1 + 1.E-20)
C
        J = 1
C
        DO 640 JC=1,NYO
C
C       DETERMINE THE LOCATION OF THE POINT YO IN TERMS OF THE SMALLEST
C       VALUE OF J SUCH THAT YO(JO) IS IN SOME RECTANGLE (I,J).
C
        YV = YC(JC)
        JJS = J+1
        IF (YV.LT.YG(JJS)) JJS=1
C
        DO 100 JJ=JJS,NYG
        IF (YV.LT.YG(JJ+1)) GO TO 120
  100 CONTINUE
C
        J = NYG
        GO TO 140
  120 J = JJ-1
  140 JD = 3
        IF (J.GE.1) GO TO 160
        JD = 0
        J = 1
        GO TO 180
  160 IF (J.LT.NYG) GO TO 180
        JD = 6
  180 DY = YG(J+2)-YG(J+1)
        I = 1
C
        DO 620 IO=1,NXO
C
C       DETERMINE THE LOCATION OF THE POINT XO IN TERMS OF THE SMALLEST
C       VALUE OF I SUCH THAT XO(IO) IS IN THE RECTANGLE (I,J).
C
        IIS = I+1
        XV = XC(IC)
        IF (XV.LT.XG(IIS)) IIS=1
C
        DO 200 II=IIS,NXG
```

```
      IF (XV.LT.XG(II+1)) GO TO 220
  200 CONTINUE
C
      I = NXG
      GO TO 240
  220 I = II-1
  240 ID = 2
      IF (I.GE.1) GO TO 260
      ID = 1
      I = 1
      GO TO 280
  260 IF (I.LT.NXG) GO TO 280
      ID = 3
  280 DX = XG(I+2)-XG(I+1)
      KD = ID+JD
      GO TO (300,360,300,440,520,440,300,360,300), KD
C
C     THIS IS FOR (XO(IO),YO(JO)) POINTS IN A SINGLE RECTANGLE (I,J)
C
  300 FV = 0.
      IJ = (J-1)*NXG+I
      IAL = 3*IJ-2
      LMAX = NP(IJ+1)-NP(IJ)
      DXA = XG(I+2)-XG(I)
      DYA = YG(J+2)-YG(J)
      XVD = (XV-XG(I))/DXA
      YVD = (YV-YG(J))/DYA
C
      DO 320 L=1,LMAX
      MPS = NP(IJ)+L-1
      KI = MP(MPS)
      XKI = (XI(KI)-XG(I))/DXA
      YKI = (YI(KI)-YG(J))/DYA
  320 FV = FV+AB(MPS)*PHI(XKI,YKI,XVD,YVD)
C
  340 FV = FV + AL(IAL) + AL(IAL+1)*XVD + AL(IAL+2)*YVD
      GO TO 620
C
C     THIS IS FOR XO(IO),YO(JO)) POINTS WHICH ARE IN TWO RECTANGLES,
C     (I,J) AND (I+1,J).
C
  360 DYA = YG(J+2)-YG(J)
      YVD = (YV-YG(J))/DYA
C
      DO 420 IP=1,2
      FC(IP) = 0.
      IS = I+IP-1
      IJ = (J-1)*NXG+IS
```

```
      IAL = 3*IJ-2
      DXA = XG(IS+2)-XG(IS)
      XVD = (XV-XG(IS))/DXA
      LMAX = NP(IJ+1)-NP(IJ)
C
      DO 380 L=1,LMAX
      MPS = NP(IJ)+L-1
      KI = MP(MPS)
      XKI = (XI(KI)-XG(IS))/DXA
      YKI = (YI(KI)-YG(JS))/DYA
  380 FC(IP) = FC(IP)+AB(MPS)*PHI(XKI,YKI,XVD,YVD)
C
  400 FC(IP)=FC(IP)+AL(IAL)+AL(IAL+1)*XVD+AL(IAL+2)*YVD
  420 CONTINUE
C
      WI = H3((XV-XG(I+1))/DX)
      FV = FC(1)*WI+(1.-WI)*FC(2)
      GO TO 620
C
C     THIS IS FOR (XO(IO),YO(JO)) POINTS WHICH ARE IN TWO RECTANGLES,
C     (I,J) AND (I,J+1).
C
  44C DXA = XG(I+2)-XG(I)
      XVD = (XV-XG(I))/DXA
C
      DO 500 JP=1,2
      FC(JP) = 0.
      JS = J+JP-1
      IJ = (JS-1)*NXG+I
      IAL = 3*IJ-2
      DYA = YG(JS+2)-YG(JS)
      YVD = (YV-YG(JS))/DYA
      LMAX = NP(IJ+1)-NP(IJ)
C
      DO 460 L=1,LMAX
      MPS = NP(IJ)+L-1
      KJ = MP(MPS)
      XKJ = (XI(KJ)-XG(I))/DXA
      YKJ = (YI(KJ)-YG(JS))/DYA
  460 FC(JP) = FC(JP)+AB(MPS)*PHI(XKJ,YKJ,XVD,YVD)
C
  480 FC(JP)=FC(JP)+AL(IAL)+AL(IAL+1)*XVD+AL(IAL+2)*YVD
  500 CONTINUE
C
      UJ = H3((YV-YG(J+1))/DY)
      FV = FC(1)*UJ+(1.-UJ)*FC(2)
      GO TO 620
C
```

```
C     THIS IS FOR (XO(IO),YO(JO)) POINTS WHICH ARE IN FOUR RECTANGLES,
C     (I,J), (I+1,J), (I,J+1), AND (I+1,J+1).
C
  520 KFC = 0
C
      DO 600 JP=1,2
      JS = J+JP-1
      DYA = YG(JS+2)-YG(JS)
      YVD = (YV-YG(JS))/DYA
C
      DO 580 IP=1,2
      IS = I+IP-1
      IJ = (JS-1)*NXG+IS
      IAL = 3*IJ-2
      KFC = KFC+1
      FC(KFC) = 0.
      DXA = XG(IS+2)-XG(IS)
      XVD = (XV-XG(IS))/DXA
      LMAX = NP(IJ+1)-NP(IJ)
C
      DO 540 L=1,LMAX
      MPS = NP(IJ)+L-1
      KI = MP(MPS)
      XKI = (XI(KI)-XG(IS))/DXA
      YKI = (YI(KI)-YG(JS))/DYA
  540 FC(KFC) = FC(KFC)+AB(MPS)*PHI(XKI,YKI,XVD,YVD)
C
  560 FC(KFC)=FC(KFC)+AL(IAL)+AL(IAL+1)*XVD+AL(IAL+2)*YVD
  580 CONTINUE
C
  600 CONTINUE
C
      WI = H3((XV-XG(I+1))/DX)
      UJ = H3((YV-YG(J+1))/DY)
      FV = WI*(UJ*FC(1)+(1.-UJ)*FC(3))+(1.-WI)*(UJ*FC(2)+(1.-UJ)*FC(4))
  620 FO(IO,JO) = FV
C
  640 CONTINUE
C
      RETURN
      END
      SUBROUTINE GRID(X,N,NX,XG,T)
C     THIS SUBROUTINE PLACES A SET OF INTERVALS OVER THE SET OF POINTS
C     (X(I), I=1,...,N).  THIS IS DONE BY PLACING APPROXIMATELY EQUAL
C     NUMBERS OF THEM WITHIN EACH INTERVAL.
C
C     THE ARGUMENTS ARE AS FOLLOWS.
C
```

```
C          N    - INPUT.  THE NUMBER OF POINTS IN THE ARRAY X.
C          X    - INPUT.  THE ARRAY OF X POINTS.
C          NX   - INPUT.  THE DESIRED NUMBER OF INTERVALS.
C          XG   - OUTPUT.  THE COORDINATES OF THE INTERVAL ENDPOINTS.
C          T    - WORK ARRAY OF DIMENSION AT LEAST N.
C
      DIMENSION X(1),XG(1),T(1)
C
      DO 100 I=1,N
  100 T(I) = X(I)
C
      CALL VSRTA(T,N)
C
      FINC = FLOAT(N-1)/FLOAT(NX+1)
  120 DO 140 J=1,NX
      FK = J*FINC + 1.
      K = FK
      WK1 = FK - K
  140 XG(J+1) = (1. - WK1)*T(K) + WK1*T(K+1)
C
      XG(1) = T(1)
      XG(NX+2) = T(N)
C
      RETURN
      END
      SUBROUTINE DECOMP(NDIM,N,A,COND,IPVT,WORK)
      REAL A(NDIM,N),WORK(N)
      INTEGER IPVT(N)
      IPVT(N) = 1
      IF(N.EQ.1) GO TO 80
      NM1 = N - 1
      ANORM = 0.
      DO 10 J=1,N
         T = 0.
         DO 5 I=1,N
            T = T + ABS(A(I,J))
    5    CONTINUE
         IF(T.GT.ANORM)ANORM = T
   10 CONTINUE
C
      DO 35 K=1,NM1
         KP1 = K + 1
         M = K
         DO 15 I=KP1,N
            IF(ABS(A(I,K)).GT.ABS(A(M,K)))M = I
   15    CONTINUE
         IPVT(K) = M
         IF(M.NE.K)IPVT(N) = -IPVT(N)
```

```
      T = A(M,K)
      A(M,K) = A(K,K)
      A(K,K) = T
      IF(T.EQ.0.)GO TO 35
      DO 20 I=KP1,N
         A(I,K) = -A(I,K)/T
20    CONTINUE
      DO 30 J=KP1,N
         T = A(M,J)
         A(M,J) = A(K,J)
         A(K,J) = T
         IF(T.EQ.0.)GO TO 30
         DO 25 I=KP1,N
            A(I,J) = A(I,J) + A(I,K)*T
25       CONTINUE
30    CONTINUE
35 CONTINUE
   DO 50 K=1,N
      T = 0.
      IF(K.EQ.1)GO TO 45
      KM1 = K - 1
      DO 40 I=1,KM1
         T = T + A(I,K)*WORK(I)
40    CONTINUE
45    EK = 1.
      IF(T.LT.0.)EK = -1.
      IF(A(K,K).EQ.0.)GO TO 90
      WORK(K) = -(EK + T)/A(K,K)
50 CONTINUE
   DO 60 KB=1,NM1
      K = N - KB
      T = 0.
      KP1 = K + 1
      DO 55 I=KP1,N
         T = T + A(I,K)*WORK(K)
55    CONTINUE
      WORK(K) = T
      M = IPVT(K)
      IF(M.EQ.K)GO TO 60
      T = WORK(M)
      WORK(M) = WORK(K)
      WORK(K) = T
60 CONTINUE
   YNORM = 0.
   DO 65 I=1,N
      YNORM = YNORM + ABS(WORK(I))
65 CONTINUE
   CALL SOLVE(NDIM,N,A,WORK,IPVT)
```

```
      ZNORM = 0.
      DO 70 I=1,N
         ZNORM = ZNORM + ABS(WORK(I))
   70 CONTINUE
      COND = ANORM*ZNORM/YNORM
      IF(COND.LT.1.)COND = 1.
      RETURN
   80 COND = 1.
      IF(A(1,1).NE.0.)RETURN
   90 COND = 1.E32
      RETURN
      END
      SUBROUTINE SOLVE(NDIM,N,A,B,IPVT)
      INTEGER IPVT(N)
      REAL A(NDIM,N),B(N)
      IF(N.EQ.1)GO TO 50
      NM1 = N - 1
      DO 20 K=1,NM1
         KP1 = K + 1
         M = IPVT(K)
         T = B(M)
         B(M) = B(K)
         B(K) = T
         DO 10 I=KP1,N
            B(I) = B(I) + A(I,K)*T
   10    CONTINUE
   20 CONTINUE
      DO 40 KB=1,NM1
         KM1 = N - KB
         K = KM1 + 1
         B(K) = B(K)/A(K,K)
         T = - B(K)
         DO 30 I=1,KM1
            B(I) = B(I) + A(I,K)*T
   30    CONTINUE
   40 CONTINUE
   50 B(1) = B(1)/A(1,1)
      RETURN
      END
C    IMSL ROUTINE NAME   - VSRTA
C
C-----------------------------------------------------------------------
C
C    COMPUTER            - IBM/SINGLE
C
C    LATEST REVISION     - JANUARY 1, 1978
C
C    PURPOSE             - SORTING OF ARRAYS BY ALGEBRAIC VALUE
```

```
C
C    USAGE                - CALL VSRTA (A,LA)
C
C    ARGUMENTS     A      - ON INPUT, A CONTAINS THE ARRAY TO BE SORTED.
C                           ON OUTPUT, A CONTAINS THE SORTED ARRAY.
C                  LA     - INPUT VARIABLE CONTAINING THE NUMBER OF
C                           ELEMENTS IN THE ARRAY TO BE SORTED.
C
C    PRECISION/HARDWARE   - SINGLE/ALL
C
C    REQD. IMSL ROUTINES  - NONE REQUIRED
C
C    NOTATION             - INFORMATION ON SPECIAL NOTATION AND
C                           CONVENTIONS IS AVAILABLE IN THE MANUAL
C                           INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C    COPYRIGHT            - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C    WARRANTY             - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                           APPLIED TO THIS CODE.  NO OTHER WARRANTY,
C                           EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----------------------------------------------------------------------
C
      SUBROUTINE VSRTA  (A,LA)
C                                     SPECIFICATIONS FOR ARGUMENTS
      INTEGER            LA
      REAL               A(LA)
C                                     SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER            IU(21),IL(21),I,M,J,K,IJ,L
      REAL               T,TT,R
C                                     FIRST EXECUTABLE STATEMENT
      M=1
      I=1
      J=LA
      R=.375
      IF (LA.LE.0) RETURN
   10 IF (I .EQ. J) GO TO 55
   15 IF (R .GT. .5898437) GO TO 20
      R=R+3.90625E-2
      GO TO 25
   20 R=R-.21875
   25 K=I
C                                     SELECT A CENTRAL ELEMENT OF THE
C                                     ARRAY AND SAVE IT IN LOCATION T
      IJ=I+(J-I)*R
      T=A(IJ)
C                                     IF FIRST ELEMENT OF ARRAY IS GREATER
```

```
C                                       THAN T, INTERCHANGE WITH T
      IF (A(I) .LE. T) GO TO 30
      A(IJ)=A(I)
      A(I)=T
      T=A(IJ)
   30 L=J
C                                       IF LAST ELEMENT OF ARRAY IS LESS THAN
C                                       T, INTERCHANGE WITH T
      IF (A(J) .GE. T) GO TO 40
      A(IJ)=A(J)
      A(J)=T
      T=A(IJ)
C                                       IF FIRST ELEMENT OF ARRAY IS GREATER
C                                       THAN T, INTERCHANGE WITH T
      IF (A(I) .LE. T) GO TO 40
      A(IJ)=A(I)
      A(I)=T
      T=A(IJ)
      GO TO 40
   35 IF(A(L).EQ.A(K)) GO TO 40
      TT=A(L)
      A(L)=A(K)
      A(K)=TT
C                                       FIND AN ELEMENT IN THE SECOND HALF OF
C                                       THE ARRAY WHICH IS SMALLER THAN T
   40 L=L-1
      IF (A(L) .GT. T) GO TO 40
C                                       FIND AN ELEMENT IN THE FIRST HALF OF
C                                       THE ARRAY WHICH IS GREATER THAN T
   45 K=K+1
      IF (A(K) .LT. T) GO TO 45
C                                       INTERCHANGE THESE ELEMENTS
      IF (K .LE. L) GO TO 35
C                                       SAVE UPPER AND LOWER SUBSCRIPTS OF
C                                       THE ARRAY YET TO BE SORTED
      IF (L-I .LE. J-K) GO TO 50
      IL(M)=I
      IU(M)=L
      I=K
      M=M+1
      GO TO 60
   50 IL(M)=K
      IU(M)=J
      J=L
      M=M+1
      GO TO 60
C                                       BEGIN AGAIN ON ANOTHER PORTION OF
C                                       THE UNSORTED ARRAY
```

```
55 M=M-1
   IF (M .EQ. 0) RETURN
   I=IL(M)
   J=IU(M)
60 IF (J-I .GE. 11) GO TO 25
   IF (I .EQ. 1) GO TO 10
   I=I-1
65 I=I+1
   IF (I .EQ. J) GO TO 55
   T=A(I+1)
   IF (A(I) .LE. T) GO TO 65
   K=I
70 A(K+1)=A(K)
   K=K-1
   IF (T .LT. A(K)) GO TO 70
   A(K+1)=T
   GO TO 65
   END
```

```
C PROGRAM DISP3M.FOR
C
C OWNER CESAR TADEU DE MIRANDA
C DATE 17/OCT/83
C
C THIS IS A FORTRAN PROGRAM THAT SETS AN UNIFORM GRID (256,256) IN
C A FORMAT SUITABLE TO BE DISPLAYED IN A IMAGE DEVICE, COMTAL. THE
C IMAGE IS EXPANDED BY A FACTOR OF FOUR.
C
C INPUT PARAMETERS
C     FILE FOR031.DAT: FILE CONTAINING THE UNIFORM GRID REPRESENTING
C     THE IMAGE.
C OUTPUT PARAMETERS
C     FILE RITA.DAT: FILE SUITABLE TO BE DISPLAYED IN THE "COMTAL".
C
C VARIABLE DECLARATION
      BYTE ME(512)
      INTEGER IPROV(256,256),I,J,K,IVALUE,ILINE,IST,IEND
C
C
C READ THE MATRIX FROM FILE IR
      TYPE *,'ENTER DESIRED INPUT UNIT'
      ACCEPT *,IR
      TYPE *,'START READING MATRIX'
      DO I=1,256
        READ(IR,130) ILINE
        DO K=0,7
          IST=32*K +1
          IEND=IST+31
          READ(IR,128)(IPROV(I,J),J=IST,IEND)
        END DO
      END DO
C
C
C TRANSPOSITION
      TYPE *,'START TRANSPOSITION'
      DO I=1,256
       DO J=1,I
        IF(I.NE.J) THEN
          ITEMP=IPROV(I,J)
          IPROV(I,J)=IPROV(J,I)
          IPROV(J,I)=ITEMP
        END IF
       END DO
      END DO
      TYPE *,'END OF TRANSPOSITION'

C
```

```
C OUTPUT THE IMAGE
      OPEN(UNIT=2,NAME='RITA.DAT',TYPE='NEW',ACCESS='DIRECT',
     1 RECORDTYPE='FIXED',RECORDSIZE=128)
      DO I=1,511,2
        IEX=I/2+1
        DO J=1,511,2
          JEX=J/2+1
          IVALUE=IPROV(IEX,JEX)
          IF(IVALUE.LE.127) THEN
            MB(J)=IVALUE
          ELSE
            MB(J)= -256+IVALUE
          END IF
          MB(J+1)=MB(J)
        END DO
        WRITE(2'I) MB
        WRITE(2'I+1) MB
      END DO
      CLOSE(UNIT=2)
      TYPE *,'END OF PROGRAM'
      STOP
128   FORMAT(3214)
130   FORMAT(I4)
132   FORMAT(I5,F10.5)
      END
```

```
C PROGRAM RMSER.FOR
C
C CESAR 2/NOV/83
C
C THIS IS A FORTRAN PROGRAM THAT COMPUTES THE RMS AND RELATIVE ERROR
C BETWEEN THE THE ORIGINAL IMAGE AND THE RECONSTRUCTED ONE.
C
C INPUT PARAMTERS
C      FILE FOR031.DAT: FILE CONTAINING THE RECONSTRUCTED IMAGE
C      REPRESENTED BY A N UNIFORM (256,256) GRID.
C      FILE IMAG1.DAT: FILE CONTAINING THE ORIGINAL IMAGE REPRESENTED
C      BY AN UNIFORM (256,256) GRID.
C
C OUTPUT PARAMETERS
C      ERROR (RMSE) AND RELATIVE ERROR.
C
C
C VARIABLE DECLARATION
C
      INTEGER F(256,256),IM(256,256),ITA(256)
      INTEGER SAVE,ACT,I,J,ILINE,IST,IEND
      REAL ERTOT,ER2,A,B,ER2BAR,ERMS,ASUM,EPSILON,AMSE
      DATA N/256/
C
C READ THE ORIGINAL DATA
C
      TYPE *,'ORIGINAL DATA BEING READ'
      OPEN(UNIT=7,NAME='IMAG1.DAT',RECORDSIZE=256,ACCESS='DIRECT',
     1     TYPE='UNKNOWN')
      DO I=1,256
C       TYPE *,'RECORD # ',I
        READ(7'I) ITA
        DO J=1,256
          F(I,J)=ITA(J)
        END DO
      END DO
      TYPE *,'ALL RECORDS WERE READ'
C
      TYPE *,'FIRST ROW'
      WRITE(6,82)(F(1,J),J=1,256)
      TYPE *,'256 ROW'
      WRITE(6,82)(F(256,J),J=1,256)
C
C SMOOTHING OF DATA
      TYPE *,'STARTING OF SMOOTHING'
      DO I=1,256
        SAVE=F(I,2)
        F(I,2)=(F(I,1)+F(I,2)+F(I,3))/3.
```

```
            DO J=3,255
              ACT=F(I,J)
              F(I,J)=(SAVE+ACT+F(I,J+1))/3.
              SAVE=ACT
            END DO
          END DO
          DO J=1,256
            SAVE=F(2,J)
            F(2,J)=(F(1,J)+F(2,J)+F(3,J))/3.
            DO I=3,255
              ACT=F(I,J)
              F(I,J)=(SAVE+ACT+F(I+1,J))/3.
              SAVE=ACT
            END DO
          END DO
          TYPE *,'END SMOOTHING'
C
C         INPUT THE APPROXIMATED DATA
          TYPE *,'START READING THE APPROXIMATED MATRIX'
          DO I=1,256
            READ(31,130) ILINE
            DO K=0,7
              IST=32*K +1
              IEND=IST+31
              READ(31,128)(IM(I,J),J=IST,IEND)
            END DO
          END DO
          TYPE *,'FINISH INPUT OF APPROXIMATED DATA'
C
C TRANSPOSITION OF THE MATRIX
          TYPE *,'START TRANSPOSITION'
          DO I=1,256
            DO J=1,I
              IF(I.NE.J) THEN
                ITEMP=IM(I,J)
                IM(I,J)=IM(J,I)
                IM(J,I)=ITEMP
              END IF
            END DO
          END DO
          TYPE *,'END TRANSPOSTION'
C
C FIND THE RMS ERROR
          TYPE *,'EVALUATE THE RMS ERROR'
          ERTOT=0.
          ASUM=0.
          DO I=1,256
            DO J=1,256
```

```
            A=FLOAT(F(I,J))
            B=FLOAT(IM(I,J))
            ER2=(A-B)**2
            ERTOT=ERTOT+ER2
            ASUM=ASUM+A**2
          END DO
        END DO
        ER2BAR=ERTOT/(FLOAT(N)**2)
        ERMS=SQRT(ER2BAR)
        TYPE *,'ERROR RMS...',ERMS
        EPSILON=ERTOT/ASUM
        AMSE=SQRT(EPSILON)*100
        TYPE *,'RELATIVE MSE...',AMSE,' %'
        TYPE *,'END OF PROGRAM'
        STOP
 82     FORMAT(1X,16I6/)
128     FORMAT(32I4)
130     FORMAT(I4)
        END
```

LIST OF REFERENCES

1. Gonzales, R.C. and Wintz, P., Digital Image Processing, Addison-Wesely, 1976.

2. Mueller, W. and Hunn, W., "Texture Analyzer System," Industrial Research, pp. 49-54, November 1974.

3. Hsi, P.C., Spline Functions and Their Applications in Digital Image Processing, Ph.D. Thesis, Syracuse University, April 1982.

4. Andrews, H.C., "Image Approximation by Variable Knot Bicubic Splines," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAM 1-3, no. 3, pp. 299-310, May 1981

5. Davis, J.C. and McCullagh, M.J., Display and Analysis of Spatial Data, Wiley, 1975.

6. K.U. Leuven Department of Computer Science Report TW55, Algorithm for Smoothing Data with Periodic and Parametric Splines, by P. Dierckx, August 1981.

7. Gerald, C.F., Applied Numerical Analysis, Addison-Wesley, 1978.

8. Naval Postgraduate School Report NPS55-80-022, Some Utility Program for the STAR Simulation Model, by J.K. Hartman, June 1980.

9. Snyder, W.V., "Algorithm 531, Contour Plotting [J6]," ACM Transactions on Mathematical Software, v. 4, no. 3, pp. 290-294, September 1978.

10. De Boor, C., A Practical Guide to Splines, Springer-Verlag, 1978.

11. Dierckx, P., "Algorithms for Smoothing Data with Periodic and Parametric Splines," Computer Graphics and Image Processing, v. 20, pp. 171-184, 1982.

12. Spath, H., Spline-Algorithmen zur Konstruktion glatter Kurven und Flachen, R. Oldenbourg Verlag, 1973.

13. Dierckx, P., "An Algorithm for Smoothing, Differentiation and Integration of Experimental Data Using Spline Functions," Journal of Computation and Applied Mathematics I, no. 3, pp. 165-184, 1975.

14. Martin, R.S. and Wilkinson, J.H., "Symmetric Decomposition of Positive Definite Band Matrices," Num. Math., v. 7, pp. 355-361, 1965.

15. K.U. Leuven Department of Computer Science Report TW54, An Improved Algorithm for Curve Fitting with Spline Functions, by P. Dierckx, July 1981.

16. Rogers, D.F. and Adams, J.A., Mathematical Elements for Computer Graphics, McGraw-Hill, 1976.

17. Akima, H., "A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points," ACM Transactions on Mathematical Software, v. 4, no. 2, pp. 148-159, June 1978.

18. Lawson, C.L., "Software for C^1 Surface Interpolation," Mathematical Software III, pp. 161-194, 1977.

19. Naval Postgraduate School Report NPS-53-81-002, Smooth Interpolation of Scattered Data by Local Thin Plate Splines, by R. Franke, 1981.

20. Franke, R., "Smooth Interpolation of Scattered Data by Local Thin Plate Splines," Comp. & Maths. with Appls., v. 8, no. 4, pp. 273-281, 1982.

21. Renka, R.J., Triangulation and Bivariate Interpolation for Irregularly Distributed Data Points, Ph.D. Dissertation, The University of Texas at Austin, 1981.

22. K.U. Leuven Applied Maths. & Prog. Div. Report TW53, A Fast Algorithm for Smoothing Data on a Rectangular Grid While Using Spline Functions, by P. Dierckx, 1980.

23. Harder, R.L. and Desmarais, R.N., "Interpolation Using Surface Splines," J. Aircraft, v. 9, pp. 189-191, 1972.

24. University of Grenoble Report 231, Fonctions-Spline du Type Plaque Mince en Dimencion 2, by J. Duchon, 1975.

25. Franke, R., Subroutine PWLOT, Naval Postgraduate School, Oct. 1983.

26. Frendenhall, G.I and Behrend, W.L., "Picture Quality - Procedures for Evaluation Subjective Effects of Interference," Proc. IRE, v. 48, pp. 1030-1034, 1960.

27. Reinsch, C.H., "Smoothing by Spline Functions," Numer. Math. v. 10, pp. 177-183, 237, 242, 248, 1967.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943	2
3.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93943	1
4.	Dr. Chin-Hwa Lee, Code 62Le Department of Electrical Engineering Naval Postgraduate School Monterey, California 93943	10
5.	Cesar Tadeu de Miranda Rua Esperanca, 196 apto. 51 12.200 - Sao Jose dos Campos - SP BRASIL	4
6.	Dr. Luiz Miranda Rua Dr. Nono Cancado, 81 35.650 - Pitangui - MG BRASIL	1
7.	Comando Geral do Pessoal (COMGEP) Diretoria de Ensino Subdiretoria Tecnica de Ensino Avenida Marechal Camara, 233 - Andar 7 20.020 - Rio de Janeiro - RJ BRASIL	1
8.	Departamento de Pesquisa e Desenvolvimento Anexo Ministerio da Aeronautica, Andar 3 Esplanada dos Ministerios, Bloco M 70.045 - Brasilia - DF BRASIL	1
9.	Centro Tecnico Aeroespacial Instituto de Atividades Espaciais Divisao de Sistemas Belicos 12.200 - Sao Jose dos Campos - SP BRASIL	1

210172

Thesis
D2992 De Miranda
c.1 Image data compression using uneven knots selection.